UACM

Universidad Autónoma
de la Ciudad de México

Nada humano me es ajeno

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN INGENIERÍA DE SOFTWARE

Implementación de un clúster tipo Beowulf para el análisis de datos de una red de estaciones meteorológicas

TESIS
PARA OBTENER EL TÍTULO DE

LICENCIADO EN INGENIERÍA DE SOFTWARE

PRESENTA:
JOSÉ PABLO HERNÁNDEZ LÓPEZ

DIRECTOR:
MTRO. ENRIQUE CRUZ MARTÍNEZ

Ciudad de México, diciembre de 2018

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

Con todo mi cariño para mis padres:
Juana de Jesús López Sepúlveda
Juan Hernández López

Agradecimientos

Quiero agradecer principalmente a mi familia por su gran apoyo incondicional en el transcurso de mi vida y de toda mi carrera profesional.

Al M. en C. Enrique Cruz Martínez, por otorgarme su confianza y apoyo desde un principio para ser partícipe de los proyectos realizados en el Laboratorio de Computo para la Enseñanza de la Ciencia (LACECI) del Plantel San Lorenzo Tezonco, así también por aceptar ser mi director de tesis, asesorándome en cada momento.

Al M. en C. Agustín González Villanueva por haber impulsado la creación del proyecto de las estaciones meteorológicas, quien lo presentó con nombre "Dispersión de micropartículas contaminantes en la zona del Plantel San Lorenzo Tezonco" a la Secretaría de Ciencia, Tecnología e Innovación de la Ciudad de México (SECITI) con clave PI2014-8, para obtener el financiamiento del mismo.

Al Ing. Fernando Robles Morales, Subdirector de Servicios de HPC en la Benemérita Universidad Autónoma de Puebla (BUAP), por su increíble apoyo y asesoría en la configuración del *Cluster Beowulf* de esta tesis, quien sin su ayuda no habría sido posible.

A la Lic. Nora Liliana Flores Salinas, quien apoyó enormemente en los tramites de este trabajo y del examen profesional.

Y finalmente a mis compañeros de laboratorio Angelica Villagómez y Sergio Galicia quienes estuvieron presentes ayudando al armado y configuración del cluster y las estaciones meteorológicas; así como a los demás compañeros de LACECI, que me ayudaron directa o indirectamente en la elaboración de estos proyectos.

A Todos Ustedes, Gracias.

Índice general

Índice de figuras

Índice de tablas

Introducción

Este trabajo recepcional fue inspirado al querer saber cómo se puede analizar un gran conjunto de datos numéricos, de tal manera que cualquier investigador que busque la forma de hacerlo pueda lograrlo con recursos mínimos, es decir, de una forma económica. Para ello se usarán tecnólogas de "Código Abierto" y de "*Hardware* Abierto" (*Open Source* y *Open Hardware*, en inglés), con la finalidad de contribuir con la comunidad científica y de desarrollo de tecnologías de la información.

Si bien el problema es ¿cómo podemos analizar un conjunto masivo de datos? y ¿qué dispositivos o infraestructura necesitamos para lograrlo? Mostraremos a lo largo de este trabajo cómo responder estas preguntas y llevarlo a cabo.

El objetivo de este trabajo es el diseño, implementación y análisis de un conjunto de Estaciones Meteorológicas para el monitoreo de un *campus* universitario; fomentando, con el uso de "herramientas libres", la replicación del trabajo en distintos lugares de la República Mexicana o del mundo, creando de esta manera una red nacional de estaciones meteorológicas y apoyar a los entusiastas del tema al desarrollo de su propia red de monitoreo ambiental proporcionando el conocimiento adecuado para lograrlo.

En este material no es posible cubrir todas las áreas que se requieren para el análisis o minado de datos, dado que para ello se han de aplicar diversas técnicas de análisis, visualización, algoritmos, metodologías, *software* e infraestructura que permitan lograr dichos objetivos llegando a ser éstos muy costosos. Por lo cual, principalmente este trabajo está enfocado en otorgar las herramientas básicas necesarias para llevar a cabo el desarrollo y creación de un *cluster* de computadoras

tipo *Beowulf* desde cero y del *software* necesario para implementar los algoritmos de minado de datos, que requiera análisis masivo de información.

El presente trabajo está organizado de la siguiente manera:

En el Capítulo 1 se detalla de forma precisa cómo construir un *cluster* tipo *Beowulf*, es decir, una red de computadoras de bajo nivel (con procesadores de uno a dos núcleos), para emular una súper computadora de varios núcleos (cincuenta, cien o más). Logrando de esta manera tener una súper computadora de bajos recursos.

En el Capítulo 2 abordamos cómo se pueden construir dispositivos electrónicos de monitoreo, en nuestro caso de monitoreo ambiental. Con tecnología embebida Arduino, la cual tanto *hardware* como *software* es libre, esto implica que se puede acceder al código y modelo raíz para su replicación e implementación independiente. Aunado a la ventaja de la gran documentación proporcionada por la comunidad de entusiastas del *software* libre.

En el Capítulo 3 se verá la logística para la planeación y distribución de las estaciones meteorológicas dentro del plantel San Lorenzo Tezonco de la UACM, con la finalidad de empezar a obtener los registros que contendrán nuestros datos a analizar.

En el Capítulo 4 veremos las herramientas de *software* necesarias para el análisis de los datos recabados, siendo estas muy potentes para el cálculo matemático, estadístico, gráfico y de generación de bases de datos. Estas herramientas también cuentan con licencia de distribución de *software* libre, para su descarga e implementación gratuita.

Finalmente, las conclusiones a las cuales se llegaron dentro del desarrollo de toda la investigación y trabajo escrito, estarán en el Capítulo 5.

Capítulo 1

Instalación y configuración del *Cluster Beowulf*

Un *cluster* es una de las herramientas, en el ámbito de la computación, más poderosas en el mundo. El término *cluster* deriva del idioma inglés, que de acuerdo a estos diccionarios, significa:

- Cambridge Dictionary [1]: *A group of similar things that are close together, sometimes surrounding something* (Un grupo de cosas similares que están cerca, a veces rodeando algo).
- Oxford Dictionary [2]: *A group of similar things or people positioned or occurring closely together* (Un grupo de cosas o personas similares posicionadas o que ocurren muy juntas).

Entonces, un *cluster* es un grupo de cosas similares; muy particularmente el término *cluster* se arraigó dentro del campo de la computación para definir a un grupo de computadoras interconectadas entre sí vía alámbrica, que trabajan en conjunto e intercambian información para ejecutar programas que demandan un alto rendimiento en procesamiento y memoria, que por sí solos en una computadora personal no podrían darse a basto con sus recursos, pero que por otro lado dentro de un *cluster* son capaces de alcanzar sus objetivos.

Así un *cluster* es un conjunto de computadoras que trabajan como una única entidad (bajo un sistema de procesamiento paralelo y/o distribuido), a cada elemento del *cluster* se le conoce como nodo y

estos están interconectados en una red de área local (LAN) para el envío y recepción de mensajes. Dependiendo de la composición de los nodos, un *cluster* puede ser [3]:

- Homogéneo: Los nodos tienen la misma configuración de *hardware* y sistema operativo.
- Heterogéneo: Los nodos tienen diferente *hardware* y sistema operativo.
- Semi-Homogéneo: Los nodos tiene similitudes entre sí ya sea de sistema operativo, *software*, o de *hardware*.

1.1. Antecedentes Históricos

La historia de las redes de computadoras comenzó en los años sesenta y setenta, prácticamente a la par del desarrollo de los primeros ordenadores digitales (computadoras), como necesidad de poder intercambiar información entre distintas ubicaciones usando estos dispositivos [4].

El Departamento de Defensa de los Estados Unidos mediante la Agencia de Investigación de Proyectos Avanzados de Defensa (DoD y DARPA por sus siglas en inglés, respectivamente) lideraron el proyecto para que se realizara la primera red de comunicaciones que interconectara los ordenadores de los distintos centros de mando y centros de investigación capaz de mantener la comunicación en una guerra nuclear. Los primeros integrantes de esta red de uso civil fueron: DARPA, la Corporación RAND (Research ANd Development) de Estados Unidos, el Instituto Tecnológico de Massachuset (MIT) y el Laboratorio Nacional de Física (NPL) del Reino Unido, junto a la Universidad de California en Los Ángeles (UCLA), que serían los desarrolladores del protocolo de transmisión TCP/IP en 1983. Todos ellos conformarían la primera red de comunicaciones denominada ARPANET.

Rápidamente se fue consolidando el concepto de "*cluster*" al hacer grupos de computadoras interconectadas y a la necesidad de procesar grandes volúmenes de datos o de ejecutar programas que requerían un

gran poder de cómputo. Gracias a Gene Amdahl de IBM (1967) [5] que publicó su teoría matemática donde describe el aceleramiento resultante de paralelizar cualquier serie de tareas, hizo posible la consolidación de los principios para una arquitectura paralela y distribuida en un *cluster*.

1.2. Configuración

La configuración del *cluster* es un proceso que se realiza por etapas, cada una de estas integra la instalación y configuración de cierto *software* que se requiere para la administración organizada de todos los recursos de los cuales va a disponer. Para empezar, se determinará el tipo de *cluster* que se ha de construir (de acuerdo a los servicios que va a ofrecer), en nuestro caso es un *cluster* para HPC (High Performance Computing) tipo *Beowulf*, es decir un armado de computadoras en red local (privada). El *cluster* contará con 40 nodos, para simplificar la configuración y entendimiento se puede hacer con un mínimo de tres máquinas, un maestro y dos esclavos; posteriormente se podrán adaptar los demás nodos más fácilmente realizando *scripts* que simplifiquen la tarea de instalación y configuración.

El *cluster* será de tipo semi-homogéneo, es decir que tendrá la misma configuración de sistema operativo y *hardware* para todos los nodos excepto para uno de ellos, el nodo maestro, que tendrá un *hardware* con características un poco diferentes para dar un mejor rendimiento en la administración de los recursos y los procesos. El *cluster* se construyó con computadoras que tienen las siguientes características importantes, cada máquina está compuesta por:

Nodo maestro:

- Procesador Intel Pentium G645 a 2.9 GHz
  - 2 CPU Core
- Memoria RAM de 6 GB
- Disco Duro de 500 GB

Nodos esclavos:

- Procesador Intel Pentium 4 a 3.2 GHz
    - 1 CPU Core
- Memoria RAM de 2 GB
- Disco Duro de 70 GB

Los nodos que se emplearon, comparados con las máquinas actuales (2018), ya no compiten con la nueva generación de computadoras con procesadores Intel Core i7 de 4 núcleos a 3.3 GHz o procesadores AMD A10 de 10 núcleos a 4 GHz, memorias RAM de 16 a 32 GB y disco duro de 1TB o unidades de estado sólido de 500 GB. Por esta razón se decidió aprovechar estas máquinas y crear un *cluster* con ellas, de esta manera podremos tener una súper computadora, es decir, una computadora con mayor capacidad de cálculo, con las siguientes especificaciones técnicas, sí tomamos en cuenta 5 nodos esclavos y el nodo maestro:

- Número de CPU's: 7
- Memoria RAM: 16 GB
- Memoria de Almacenamiento: 850 GB

Nota: si agregamos más nodos al *cluster*, su capacidad de cálculo se irá incrementando en proporción del numero de nodos que contenga.

Este nuevo arreglo de computadoras puede efectuar operaciones más complicadas y resolver problemas que antes no se podían realizar con una sola máquina de este grupo, pero que en conjunto son capaces de eso y más haciendo uso del cómputo en paralelo y/o distribuido. Otra ventaja de este tipo de *cluster* es su escalabilidad, es decir, es posible aumentar el número de nodos en la red a 50, 100, 1000 o más; de esta forma incrementamos el poder de cómputo del *cluster*. Hay que tomar en cuenta que entre más nodos tenga, más corriente eléctrica consumirá y también generará más calor, por lo cual es importante tener una buena instalación eléctrica y un buen sistema de enfriamiento para tener en óptimas condiciones al *cluster*.

Para configurar el *cluster* se debe instalar un sistema operativo base en todos los nodos, se pueden usar distintos sistemas operativos para la configuración de un *cluster*, desde Windows, MacOS, o cualquier distribución de Linux. Una vez listos los nodos con su sistema operativo podemos proceder a las siguientes configuraciones. En este trabajo se empleó como base CentOS 7 ya que es más estable que otras distribuciones Linux al ser una distribución de Red Hat, empresa dedicada a dar soporte empresarial (*cluster* y *big data*) desde 1993 con su propia distribución *Red Hat Enterprise Linux* (RHEL), de la cual CentOS implementa varias fuentes del mismo teniendo de esta manera la compatibilidad con las librerías y el respaldo de los desarrolladores de la empresa.

1.3. Red Local de Computadoras

Después de la instalación y actualización de paquetes del propio sistema operativo, se configuran los *scripts* de las tarjetas de red de cada nodo, sólo el nodo maestro tendrá dos tarjetas de red, una para la configuración de la red interna LAN (*Local Area Network*, o red de área local), y otra para la configuración de la red externa WAN (*Wide Area Network*, o red de área amplia), los demás nodos solo tendrán una tarjeta de red para conectarse al maestro mediante la LAN. La red interna será la encargada del envío y recepción de mensajes de cada nodo dentro del *cluster* (con lo cual se podrán monitorear los procesos de los nodos esclavos); y la red externa, con la cual el *cluster* se puede conectar a internet [6].

La conexión física de la red interna LAN, o topología de red, se llevará a cabo con un *switch* que interconectará a todos los nodos en una configuración específica, *multistep* maestro/esclavo, para el envío y recepción de datos, como se muestra en la Figura 1.1. En esta topología de red cada paquete de datos que es enviado por algún nodo, pasa a través del nodo central (o maestro), el cual reenvía el paquete de regreso a toda la red para que lo reciban los demás nodos conectados a ella, pero sólo el nodo donde coincida su IP con la dirección IP del mensaje podrá leerlo.

El uso de esta topología de red es muy conveniente, su manteni-

miento y configuración es sencillo, se pueden agregar y quitar nodos fácilmente; además si un nodo dentro de la red llegara a fallar o uno de los cables se llegara a romper, sólo habría que reparar el nodo o sustituir el cable según sea el caso; en estas circunstancias la red no se ve afectada y sigue funcionando a excepción de que el nodo que falle sea el nodo maestro, en cuyo caso la red se verá interrumpida inmediatamente, por lo que sería urgente arreglar el problema y reiniciar la red.

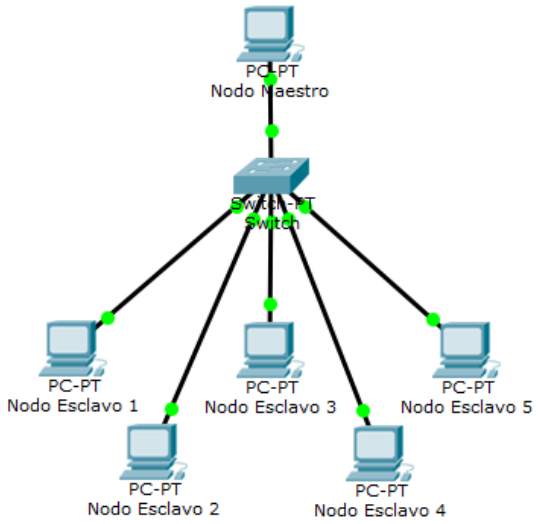

Figura 1.1: Esquema maestro/esclavos de un *cluster Beowulf*

Para poder configurar los *scripts* de red en el sistema operativo CentOS 7 tenemos que hacer dos cosas, primero identificar el nombre de nuestra tarjeta de red y segundo modificar el archivo de dicha tarjeta. Entonces para poder identificar nuestras tarjetas de red hacemos uso de la terminal y escribimos la instrucción:

- ip addres list

Esta instrucción nos mostrará una lista de los dispositivos de red que tiene la computadora, debemos de identificar un nombre similar a "enp1s0", este nombre es el de la tarjeta de red y será también parte del nombre que tendrá nuestro archivo de configuración el cual buscaremos

dentro del directorio "/etc/sysconfig/network-scripts" para poder editarlo y agregar las características faltantes para la configuración de red. Dentro de esta etapa se sugiere planear previamente la configuración de la red como se muestra en la Tabla 1.1; además es recomendable crear una copia de respaldo del archivo antes de modificarlo.

	Externa (WAN)	Interna (LAN)	Nodos (LAN)
IPADDR	192.168.123.X	10.0.X1.Y1	10.0.Xn,Yn
NETMASK	255.255.255.0	255.255.0.0	255.255.0.0
GATEWAY	192.168.123.254	10.0.X1.Y1	10.0.X1.Y1
NETWORK	192.168.123.0	10.0.0.0	null
BROADCAST	192.168.123.255	10.0.255.255	null
DNS	8.8.8.8	8.8.8.8	8.8.8.8

Tabla 1.1: Configuración de rede interna y externa

NOTA: Las direcciones de red son de ejemplo, en sus computadoras variarán de acuerdo a la red a la que estén conectados.

Dentro del archivo se modifican y/o agregan los siguientes parámetros de acuerdo al nodo correspondiente:

- BOOTPROTO = static
- IPV4_FAILURE_FATAL = yes
- IPV6INIT = no
- ONBOOT = yes
- IPADDR = (IP de acuerdo al nodo correspondiente)
- NETMASK = (de acuerdo a su configuración de red LAN)
- GATEWAY = (de acuerdo a su configuración de red LAN)
- NETWORK = (de acuerdo a su configuración de red LAN)
- BROADCAST = (de acuerdo a su configuración de red LAN)

Por último, se activan y reinician los siguientes servicios de red:

- NetworkManager y
- network.service

1.4. *Ganglia*

Ganglia es un sistema de monitoreo distribuido de ambiente gráfico, surgió como un proyecto de código abierto en la Universidad de Berkeley California, su *software* se distribuye bajo la licencia BSD (*Berkeley Software Distribution*). *Ganglia* nos permite monitorear la actividad que hay en un *cluster* y sus nodos, su *software* es bastante robusto y puede implementarse en distintos sistemas operativos y arquitecturas de procesadores para vincular hasta 2000 nodos o más en sistemas de alto rendimiento; internamente utiliza XML (*eXtensible Markup Language*), XDR (*eXternal Data Representation*) y RRDtool (*Round Robin Database tool*) para la representación, transporte y almacenamiento y visualización de datos respectivamente [7, 8].

La instalación y configuración es relativamente sencilla. Primero se crea el servidor en el nodo maestro y luego los clientes en los nodos de cálculo; para ello desde el nodo maestro navegamos en la página, http://ganglia.info/, y se descargan los paquetes *Ganglia Monitor Core* y *Ganglia Web* en alguna de sus versiones (según sea su conveniencia), posteriormente se instalan las dependencias necesarias para construir el paquete "rpm" e instalarlo con el uso de "yum install". Una vez que se haya terminado de instalar procedemos a configurar los *scripts* que identificarán a nuestro *cluster* dentro del entorno *Ganglia*, para esto editamos los siguientes archivos:

/etc/ganglia/gmetad.conf
Editar la línea:
data_source "Nombre cluster" 1 "IP del nodo maestro"

/etc/ganglia/gmond.conf
Modificar el archivo para las siguientes partes:

cluster {
name=“Nombre *cluster*”
owner=“unespecified”
latlong=“unespecified”
url=“unespecified”
}

udp_send_channel {
# bind_hostname ...
# mcast_join ...
host = “IP del nodo maestro”
port = 8649
ttl = 1
}

udp_recv_channel {
# mcast_join = 239.2.11.71
port = 8649
# bind = 239.2.11.71
# retry_bind = true
# Size of the UDP buffer. If you are handling lots of metrics you really
# should bump it up to e.g. 10MB or even higher.
# buffer = 10485760
}

tcp_accept_channel {
port = 8649
# If you want to gzip XML output
# gzip_output = no
}

(guardar y salir)

Finalmente, lo último que tenemos que hacer es habilitar y activar los siguientes servicios:

- httpd
  - Sirve para responder las solicitudes automáticamente de documentos de hipertexto y multimedia a través del protocolo HTTP (*Hypertext Transfer Protocol*)
- gmond
  - Sirve para monitorear y recopilar estadísticas en cada nodo
- gmetad
  - Sirve para realizar sondeos periódicos y almacenar las métricas en el motor de almacenamiento RRD

Para los nodos de cálculo copiamos los paquetes, rpm, generados al principio en el nodo maestro a cada nodo esclavo (a excepción del paquete ganglia-gmetad*.rpm) e instalamos; modificamos adecuadamente los archivos gmetad.conf y gmond.conf, habilitamos y activamos los servicios httpd, gmond, gmetad y ya podremos ver nuestra configuración terminada.

Accedemos a cualquier navegador de internet (desde el nodo maestro) y accedemos a la página "http://(IP del nodo maestro)/ganglia/" y podremos ver algo como se muestra en la Figura 1.2 dependiendo del número de nodos que hayamos configurado.

1.5. Sistema de Archivos en Red

Una parte muy importante dentro de la configuración de un *cluster* es el sistema de archivos de red o NFS (*Network File System*, por sus siglas en inglés), este sistema de archivos distribuido es un protocolo de nivel de aplicación de acuerdo al modelo de interconexión de sistemas abiertos OSI (*Open System Interconnection*, por sus siglas en inglés), este protocolo, NFS, permite a los usuarios-clientes (nodos esclavos) acceder de forma remota a toda clase de archivos alojados en otra

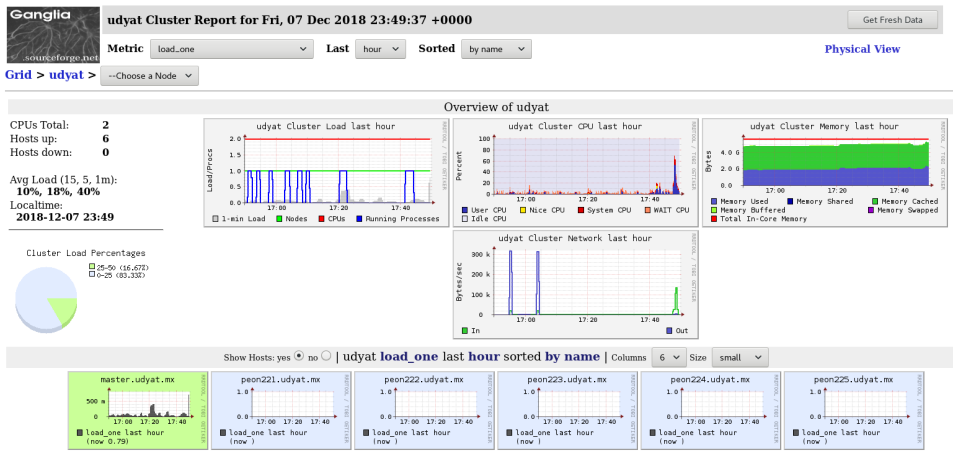

Figura 1.2: Ejemplo del Monitor *Ganglia Web*

máquina servidor (nodo maestro) dentro de la misma red, tal y como acceden a sus propios archivos locales [9].

El protocolo NFS tiene ventajas como:

- Mantener centralizada la información evitando duplicidad de ésta
- Los nodos utilizan menos espacio de disco al delegar su directorio `home` al servidor NFS, logrando con esto un acceso desde cualquier punto de la red
- Las transacciones que se realicen sobre cualquier archivo son síncronas e íntegras, es decir, si más de un usuario está trabajando con un mismo archivo, el servidor sólo puede liberar el recurso cuando la transacción por un usuario haya terminado

Para poder configurar el servicio NFS requerimos en cualquier nodo el paquete "nfs-utils" ya sea para configurar el cliente o el servidor, además de habilitar el puerto nfs para el *firewall*. Como ejemplo veamos cómo compartir el directorio /home:

- En el nodo maestro:
  - Desmontar y eliminar el directorio /home que está por default

- Nota: Genere un respaldo (imagen) de dicha unidad si tiene archivos que recuperar posteriormente.

- Crear un nuevo volumen lógico /home con una capacidad mayor en GB ya que este almacenará a todos los usuarios y sus archivos.
- Asignar su sistema de archivos a dicho volumen (de acuerdo al que maneje su sistema operativo)
- Montar el nuevo /home
- Editar el archivo (/etc/fstab) y agregar la línea:
- /dev/mapper/cl-home /home xfs defaults 0 0
  - Nota: Antes verifique la estructura de sus directorios con la instrucción: lvdisplay
- Editar el archivo (/etc/exports) y agregar la línea:
- /home 10.0.0.0/255.255.0.0(rw,async,no_root_squash)
  - Nota: La IP/NETMASK son de acuerdo a la red LAN que haya definido
- Activar y reiniciar el servicio nfs

- En los nodos esclavos:

  - Desmonta el antiguo /home
  - Editar el archivo (/etc/fstab) y agregarla línea:

    master:/home /home nfs rsize=8192,wsize=8192,timeo=14, intr 0 0

    Observaciones:

    - El valor “rsize” es el número de bytes usados cuando se lee desde el servidor.
    - El valor “wsize” es el número de bytes usados cuando se escribe en el servidor.
    - El valor predeterminado para ambos es 1024, pero si se usan valores mayores como 8192 puede mejorar el rendimiento.
    - El valor “timeo” es la cantidad de tiempo, en décimas de segundo, que se debe esperar antes de volver a enviar una transmisión después de un tiempo de espera RPC.

- La opción “intr” permite enviar señales para interrumpir las operaciones de los archivos si ocurre un tiempo de espera mayor en la partición compartida.

- Montar el nuevo /home compartido con la instrucción: mount (IP del nodo maestro):/home /home
- Activar y reiniciar el servicio nfs

1.6. Administrador de recursos TORQUE

Para continuar con nuestra configuración hacemos uso de TORQUE, un *software* gestor de recursos y colas de código abierto Terascale (o *Terascale Open-Source Resource and QUEue Manager*, por su acrónimo en inglés). TORQUE es un *software middleware* que interactúa en redes para administrar trabajos que distribuye, en varios nodos de cálculo, de acuerdo a lo que es requerido o solicitado, monitoreando a cada uno hasta que finalice su tarea; de esta manera simplifica el trabajo de los desarrolladores en la compleja tarea de generar las conexiones y sincronizaciones que son necesarias en los sistemas distribuidos [10, 11].

Para realizar la instalación seguimos los siguientes pasos:

- Descargar de la siguiente página el archivo torque-x.x.x.tar.gz e instalamos:
- http://www.adaptivecomputing.com/support/download-center/
- Copiar los siguientes servicios a nuestro directorio /etc/init.d/
  - pbs_server
  - pbs_sched
  - trqauthd
- Modificar los siguientes archivos:
  - Agregar el hostname del nodo maestro a:

    - /var/spool/torque/server_name
  - Agregar la ruta "usr/local/lib" a:
    - /etc/ld.so.conf.d/torque.conf
  - Agregar los nodos del *cluster* a:
    - /var/spool/torque/server_priv/nodes

- Activamos e iniciamos los servicios:
  - pbs_server
  - pbs_sched
  - trqauthd
- Finalmente construimos los paquetes para los nodos con la siguiente instrucción:
  - make packages
- Copiamos el directorio de TORQUE a los nodos con 'rsync' e instalamos los siguientes paquetes:
  - torque-package-mom
  - torque-package-clients
- Copiamos los archivos de servicios generados al directorio /etc/init.d/ de cada nodo, los servicios son:
  - pbs_mom
  - trqauthd
- Activamos e iniciamos los servicios:
  - pbs_mom
  - trqauthd
- Listo, podemos verificar nuestros nodos desde el maestro con:
  - pbsnodes

1.7. Administración de Usuarios

Por último lo que resta por hacer es crear a los usuarios que tendrán acceso a *cluster* para que puedan realizar y ejecutar sus programas. Para ello lo que tenemos que hacer es:

1. Crear un usuario:
   a) useradd -m nombre_usuario
2. Verificar el ID de usuario:
   a) cat /etc/passwd | grep nombre_usuario
3. Agregar al usuario en todos los nodos:
   a) pdsh "groupadd -g ID_usuario nombre_usuario"
   b) pdsh üseradd -u ID_usuario -g ID_usuario -m nombre_usuario"
4. Iniciar sesión con el nuevo usuario y generar su llave:
   a) ssh-keygen -t rsa (teclear enter en todas las opciones)
5. Copiar la llave a todos los nodos y listo:
   a) ssh-copy-id -i .ssh/id_rsa.pub peonXYZ

Nota: Si un nodo no se conecta vía ssh desde el maestro, verifique se hayan agregado adecuadamente los usuarios y sus grupos (usando cat /etc/passwd | grep nombre_usuario). De ser necesario elimine y cree de nuevo las llaves en el nodo defectuoso, copie la id_rsa.pub del maestro al nodo y agregue los usuarios y sus grupos correspondientes con sus IDs a dicho nodo.

1.8. Configuración Externa

Finalmente hemos concluido con la configuración interna del *Cluster Beowulf*. A continuación podemos ver un par de fotos de como se ha organizado externamente el *cluster* de computadoras y fueron acomodados los CPU's en estantes (Figura 1.4) para minimizar el espacio que ocupan, así como su conexión al *switch* (Figura 1.5) y un par de monitores para la administración desde el nodo maestro (Figura 1.3).

Cabe mencionar que la configuración expuesta es de forma general, tratando de enfocar los puntos más importantes de esta, dado que dependiendo del sistema operativo o versiones del *software* y paqueterías que se elijan, la configuración variará ligeramente en repositorios, archivos y rutas dentro del sistema, pero enfocándose en los puntos descritos con anterioridad siempre y cuando se basen en un sistema operativo de la familia *Linux* de *Red Hat*.

Para poder ver a detalle la configuración paso a paso, e instrucción a instrucción que se realizó en este trabajo para la configuración del *Cluster Beowulf* con sistema operativo CentOS7, diríjase al Apéndice A.

Figura 1.3: Monitores del nodo maestro

(a) Torre No.1 (b) Torre No.2

Figura 1.4: Organización de CPU's en estantes

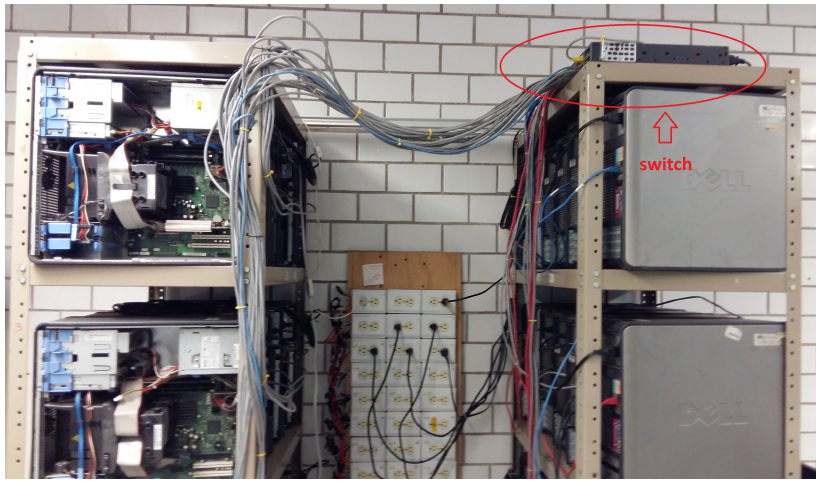

Figura 1.5: Conección de nodos al *switch*

Capítulo 2

Diseño e implementación de estaciones meteorológicas

La siguiente etapa del proyecto, es la elaboración y configuración de un conjunto de estaciones meteorológicas. Una estación meteorológica, es un dispositivo creado para medir las variables ambientales cómo: humedad, presión, temperatura, etc. que derivan del comportamiento del clima. En este trabajo se diseñó y construyó una mini estación usando tecnología embebida *Arduino*, plataforma de desarrollo de *hardware* y *software* libre, con licencia *Creative Commons*. A continuación se verá su implementación.

2.1. ¿Qué es *Arduino*?

Arduino es una plataforma de desarrollo electrónico, creada en el Instituto de Diseño de Interacción IVREA en 2005 por el investigador italiano Massimo Banzi como un proyecto académico para facilitar el desarrollo de los trabajos entre estudiantes, profesores y entusiastas de la electrónica. *Arduino* es una placa de micro controladores preensamblada que cuenta con su propio entorno de desarrollo integrado, IDE (*Integrated Development Environment*), y su propio lenguaje de programación, con *Arduino* uno puede crear distintos tipos de proyectos de electrónica gracias a su *hardware* y *software* fácil de usar [12,13].

El dispositivo *Arduino* Uno, Figura 2.1, cuenta con:

- Un microcontrolador ATmega328
- 32k bytes de Memoria Flash (5k para el gestor de arranque)
- 2k bytes de SRAM
- 1k byte de EEPROM
- 14 pines Digitales de I/O (6 salidas PWM)
- 6 entradas Análogas
- Reloj de 16MHz de velocidad.
- Voltaje de entrada 7-12V (límite de 6-20V)
- Voltaje de operación 5V
- Corriente continua en el pin 3.3V 50 mA

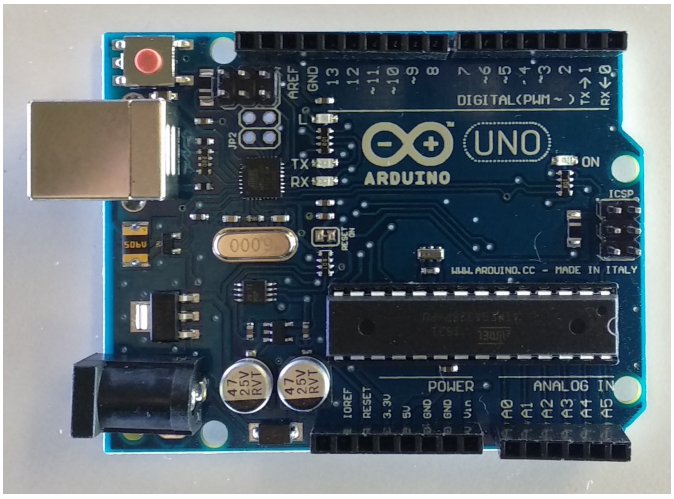

Figura 2.1: *Arduino* Uno

En la Figura 2.2 se muestra el IDE de *Arduino* que cuenta con una interfaz gráfica sencilla y fácil de usar, así como la forma en que se implementa su código, donde cuenta con dos secciones principales; la primera es el *setup()* donde se declaran e inicializan todas las variables del programa, y la segunda es el *loop()* donde se escribe el código principal de nuestro programa.

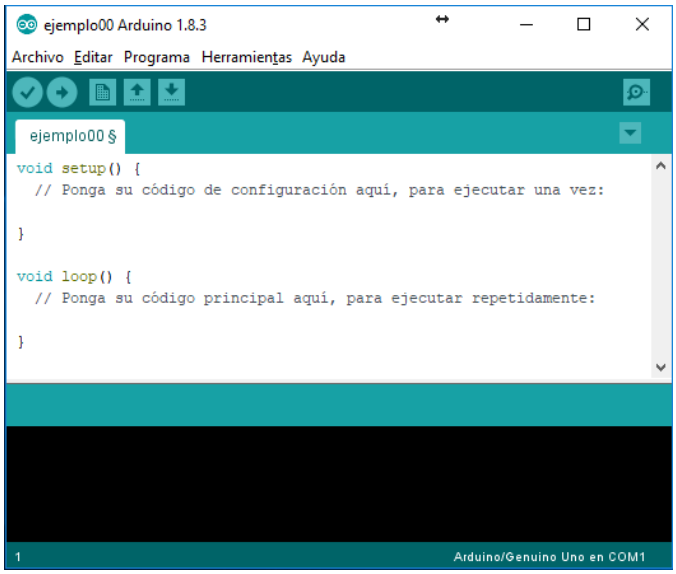

Figura 2.2: IDE *Arduino*

2.2. Descripción de la Estación Meteorológica

Como ya se mencionó, una estación meteorológica es un dispositivo electrónico que registra, mediante sensores, las variables del medioambiente donde esta esté localizada; las variables que se registran son: humedad, temperatura, presión atmosférica, densidad de polvo y velocidad de viento. Esta estación meteorológica se construyó con tecnología *Arduino* Uno y un conjunto de sensores que se describirán más adelante.

El proyecto se pensó a *priori* para analizar el clima dentro de la Universidad Autónoma de la Ciudad de México en el plantel San Lorenzo Tezonco, e incluso en otras universidades y escuelas dado que al usar tecnologías *open source* como *Arduino*, este proyecto pueda replicarse sin infringir leyes de *copyright*; ampliando así su funcionalidad como herramienta de investigación para alumnos y profesores interesados.

Para este proyecto se pensó en un conjunto de nueve estaciones meteorológicas en inicio; posteriormente se podrán incluir más estaciones a la red o mejorar estas ampliando el número y tipo de sensores. De esta manera podemos formar distintos arreglos de distribución en superficie de las estaciones meteorológicas dentro de la universidad y realizar las lecturas deseadas. Cabe mencionar que si se desea ampliar el número de sensores, se tendrá que cambiar la placa base *Arduino* Uno, dado que ésta cuenta con pocas entradas analógicas/digitales las cuales se han utilizado por completo para los sensores y tarjeta de memoria.

2.3. Sensores a utilizar

Para la elaboración de la estación meteorológica se utilizaron los siguientes materiales: Una placa *Arduino* Uno (microcontrolador ATmega328), adaptador de memoria micro SD, memoria micro SD de 8 GB, sensores DHT11, BMP180, GP2Y10 y AH3503 que sirven para tomar las lecturas de humedad, presión atmosférica, densidad de polvo y velocidad de viento respectivamente. A continuación, conoceremos su aspecto físico, características técnicas y código asociado de forma individual para después ver la implementación completa.

2.3.1. Sensor DHT11

Sensor DHT11, Figura 2.3, encargado de registrar humedad y temperatura. Sus características técnicas son [14]:

- Funciona a 3.5V y 5.5V de alimentación
- Rango de temperatura: 0°C a 50°C con +/- 2°C de precisión

- Rango de humedad: 20 % a 80 % con 5 % de precisión
- 1 muestra por segundo (1Hz)

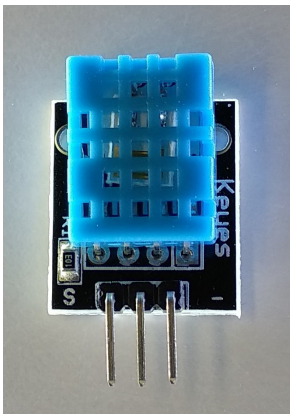

Figura 2.3: Sensor DHT11

2.3.2. Sensor BMP180

Sensor Barométrico BMP180, Figura 2.4, encargado de registrar la presión atmosférica y temperatura. Sus características técnicas son [15]:

- Interfaz digital de dos cables (I2C)
- Rango de medición: 300 – 1100hPa
- Completamente calibrado
- Alimentación: 1.8V – 3.6V
- Rango de temperatura: -40°C a 85°C con +/- 2°C de precisión
- Velocidad del protocolo máxima: 3.4 MHz.

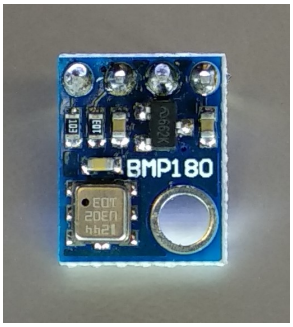

Figura 2.4: Sensor BMP180

2.3.3. Sensor Sharp GP2Y1010AU0F

Sensor de densidad de polvo Sharp GP2Y1010AU0F, Figura 2.5, encargado de detectar las partículas en el aire. Sus características técnicas son [16]:

- Dimensiones 46.0 × 30.0 × 17.6 mm
- Corriente de bajo consumo (ICC: máx. 20 mA)
- Alimentación 5V-7V
- El polvo es detectado por fotometría de un sólo pulso
- Sensibilidad de partículas: 0,1 mg / m^3
- Conector JST (Japan Solderless Terminal) de 6 pines

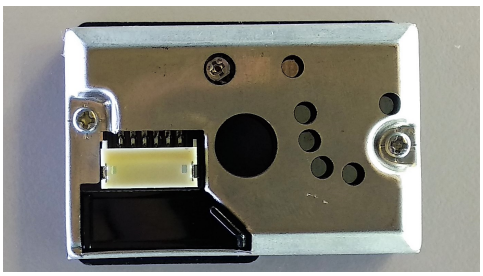

Figura 2.5: Sensor Sharp GP2Y1010AU0F

2.3.4. Sensor Magnético de Efecto Hall Ky-003

Sensor Magnético de Efecto Hall Ky-003, Figura 2.6, encargado de registrar la velocidad del viento mediante una hélice que se implementará con un par de imanes de neodimio para que el sensor pueda registrar los giros que la hélice da a causa del aire que la mueve. Sus características técnicas son [17]:

- Sensor lineal que va a traducir en tensión las variaciones del campo magnético que perciba.
- Consumo de 10 mA
- Voltaje de operación de 4.5V a 24V
- Temperatura de operación -40°C a 85°C
- Trabaja en campos magnéticos en el rango de 650-1000 Gauss.
- Dimensiones: 2.2 x 1.5 x 0.5 cm

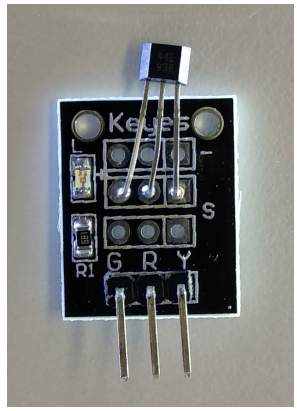

Figura 2.6: Sensor Magnético de Efecto Hall Ky-003

2.4. Sistema de Almacenamiento

El sistema de almacenamiento para la estación meteorológica se implementa con un adaptador y una tarjeta de memoria micro SD de 8 GB como se muestra en la Figura 2.7. Los datos serán almacenados en archivos de texto (*.txt) con la abreviación "ESTMET-X", donde X será el número que le corresponde a cada estación meteorológica, en este caso X=1,2,3,...,9. Los datos recolectados y generados por las estaciones meteorológicas se disponen en un arreglo de 6 columnas dentro de éste archivo de la siguiente manera: humedad, temperatura 1 (Sensor DHT11), temperatura 2 (Sensor BMP180), presión atmosférica, densidad de polvo y velocidad de viento.

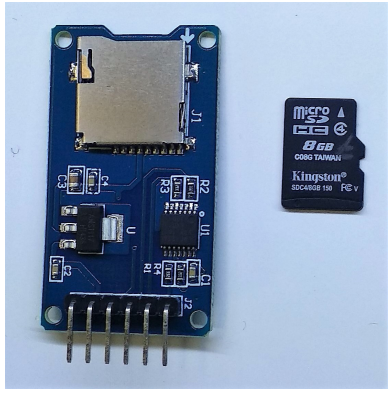

Figura 2.7: Adaptador y Tarjeta Micro SD 8 GB

2.5. Sistema de Alimentación

El sistema de alimentación para la estación meteorológica, para la placa *Arduino* Uno y por ende de todos los sensores en ella conectados, es una batería Kapton modelo SB-0604, Figura 2.8, con las siguientes especificaciones:

- Voltaje: 6VCC
- Corriente: 4.5 Amperes
- Dimensiones: 10cm x 7cm x 4.7cm

- Recargable

Figura 2.8: Batería Kapton SB-0604

2.6. Implementación de Estación Meteorológica

Después de haber realizado las pruebas individuales de cada sensor con una *protoboard* y la plataforma *Arduino* y haber constatado que funcionan correctamente, se procede a implementar todos y cada uno de ellos dentro de la misma *protoboard* y placa base *Arduino* Uno, como lo muestra la Figura 2.9.

Una vez que todos los sensores se encuentran conectados y configurados a la plataforma *Arduino* Uno correctamente, se diseña una forma de proteger los circuitos de posibles lluvias y acumulaciones excesivas de polvo, así como de una base para transportar y montar todos los componentes: la *protoboard*, sensores, dispositivo *Arduino*, sistema de alimentación, sistema de almacenamiento y hélice para el sensor de efecto Hall. La Figura 2.10 muestra el modelo final de la estación meteorológica en esta etapa.

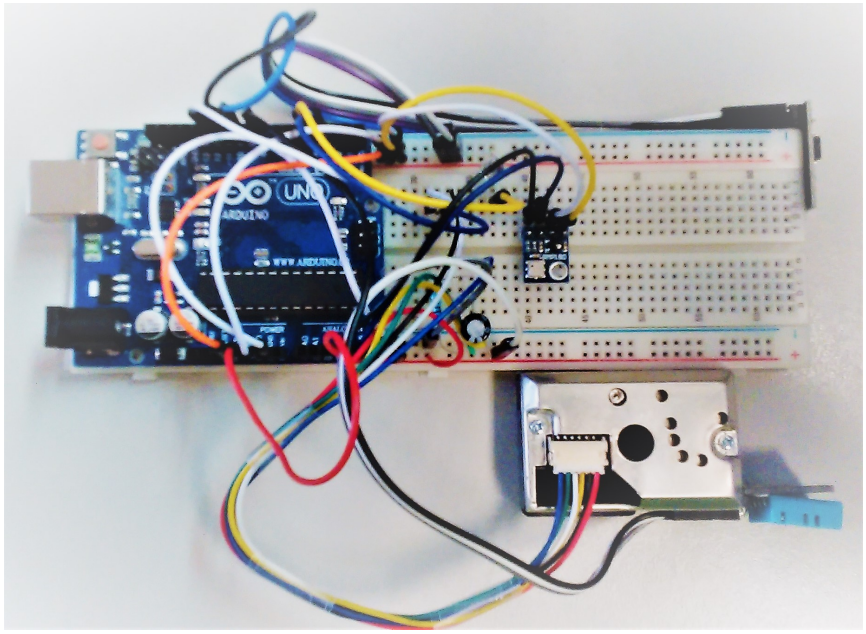

Figura 2.9: Estación Meteorológica Prototipo Versión 1

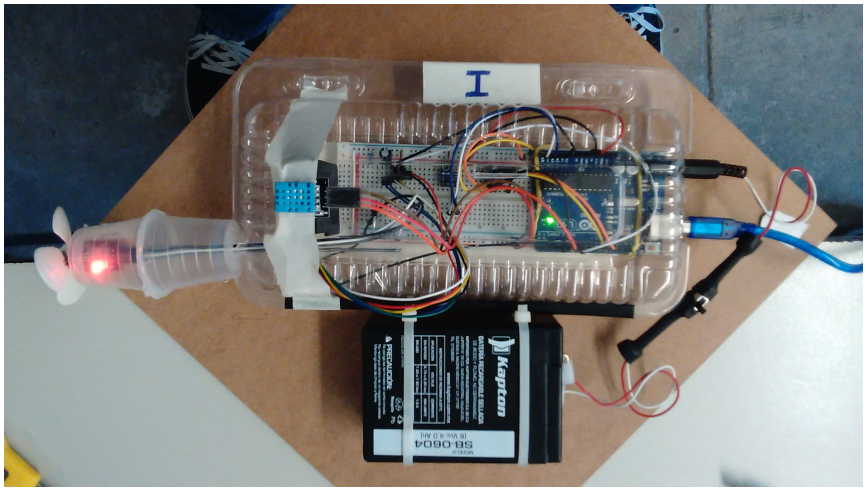

Figura 2.10: Estación Meteorológica versión final

2.7. Código de Implementación

Finalmente, el código para la implementación de todos los sensores y dispositivos conectados a la plataforma *Arduino* Uno quedará de la siguiente manera, ver el Apéndice B.1.

Capítulo 3

Análisis del área de trabajo

En esta etapa nos encargamos de realizar la planeación de la distribución de las Estaciones Meteorológicas dentro del plantel San Lorenzo Tezonco de la UACM. Para tener una mejor perspectiva del terreno y los edificios hacemos uso de *Google Maps* que nos ofrece las vistas aéreas necesarias del campus para realizar una planeación adecuada de la distribución de las estaciones dentro del mismo.

3.1. Plantel San Lorenzo Tezonco

Haciendo uso de la herramienta web de navegación vial e imagen satelital, *Google Maps* [18], analizamos y tomamos un par de capturas de pantalla que nos muestre el área de interés a trabajar con las estaciones meteorológicas, en este caso el plantel San Lorenzo Tezonco, tal como lo muestra la Figura 3.1.

Una vez realizado esto, trabajamos la imagen y cuadriculamos el mapa con la referencia de la distancia que nos proporciona la misma aplicación *web* (esquina inferior derecha, Figura 3.2).

Figura 3.1: Vista Aérea del Plantel San Lorenzo Tezonco

Figura 3.2: Referencia de distancia de 20m

Para seguir trabajando la imagen, la recortamos y ajustamos a un área de interés más reducida (Figura 3.3), aquella donde la comunidad universitaria transite habitualmente.

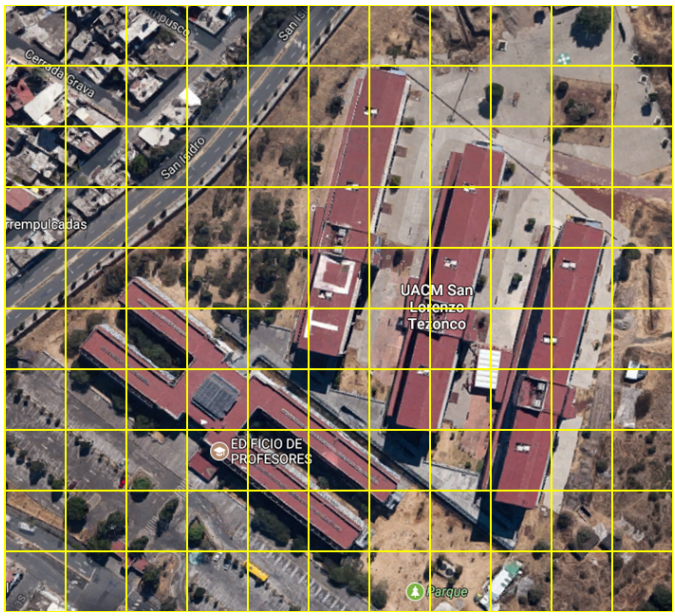

Figura 3.3: Área de interés del plantel San Lorenzo Tezonco

3.2. Distribución de las Estaciones

De acuerdo al mapa de la Figura 3.3, y tomando en cuenta que nos interesan las secciones donde las personas transitan cotidianamente, podemos destacar seis áreas dentro del *campus* donde esto se cumple. Para identificarlas las numeramos del 1 al 6 como se muestra en la Figura 3.4.

Como se tiene un total de nueve estaciones meteorológicas, generamos dos grupos de áreas:

- Grupo 1: Color Verde (del 1 al 3)
  - Edificios A, B y C donde se imparten las clases
- Grupo 2: Color Amarillo (del 4 al 6)
  - Corredores, por donde las personas van de un edificio a otro.

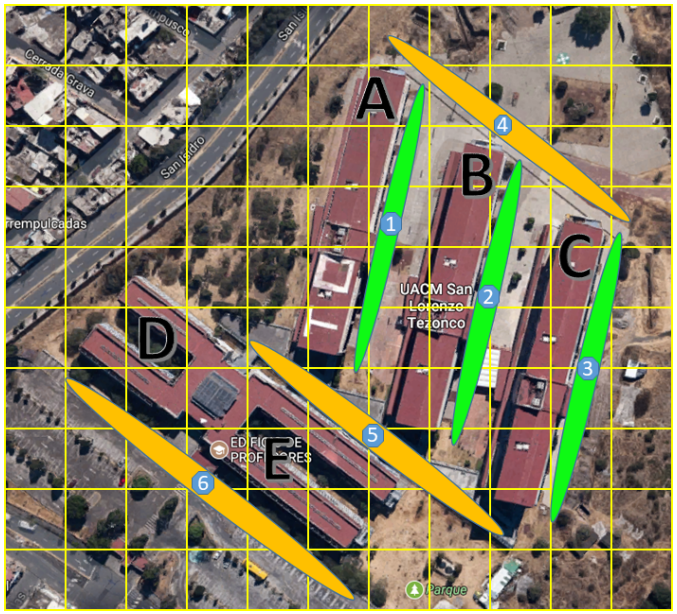

Figura 3.4: Áreas con mayor tránsito de personas

Donde cada área de ambos grupos contendrá tres estaciones meteorológicas distribuidas a lo largo de cada una para cubrirla de forma general, es decir, al centro y en sus extremos.

3.3. Recolección de Datos

Finalmente, después de colocar las estaciones en sus respectivos lugares y realizar las debidas mediciones; tenemos que cada estación meteorológica realiza una captura de datos (de las variables ambientales) cada tres segundos, lo cual implica que por cada estación estará generando 1,200 registros por hora; es decir 10,800 registros por hora de las nueve estaciones en conjunto.

Si dejamos trabajar una estación las 24 horas del día tendríamos 28,800 registros por día de cada estación, lo cual implica que tendremos 259,200 registros de las nueve estaciones en un día de trabajo.

En promedio un archivo CSV con 1200 registros pesa 85 KB (*Kilo Bytes*). La disposición de las columnas que contiene cada archivo se muestra a continuación en la Tabla 3.1.

fecha	hora	estacion	edificio	posicion	altura	hum1	tmp1	tmp2	ps	dens	frq
02/09/2017	12:00:01	e1	A	Y1	P0	40.00	21.00	22.50	78160.00	21.14	0.00
02/09/2017	12:00:04	e1	A	Y1	P0	40.00	21.00	22.50	78157.00	26.95	0.00
02/09/2017	12:00:07	e1	A	Y1	P0	40.00	21.00	22.60	78159.00	16.99	0.00
02/09/2017	12:00:10	e1	A	Y1	P0	40.00	21.00	22.60	78161.00	18.65	0.00
02/09/2017	12:00:13	e1	A	Y1	P0	40.00	21.00	22.60	78158.00	22.80	0.00
02/09/2017	12:00:16	e1	A	Y1	P0	40.00	21.00	22.60	78161.00	20.31	0.00
02/09/2017	12:00:19	e1	A	Y1	P0	40.00	21.00	22.60	78158.00	18.65	0.00
02/09/2017	12:00:22	e1	A	Y1	P0	40.00	21.00	22.60	78161.00	6.20	0.00
02/09/2017	12:00:25	e1	A	Y1	P0	40.00	21.00	22.60	78158.00	22.80	0.00

Tabla 3.1: Disposición de variables meteorológicas

Capítulo 4

Minería de datos

Para realizar el análisis e interpretación de los datos generados por las estaciones meteorológicas haremos uso de una metodología denominada "Descubrimiento de Conocimiento en Bases de Datos", o KDD (*Knowledge Discovery in Databases*) por sus siglas en inglés. Este es un proceso iterativo compuesto por varias etapas las cuales comienzan por la recolección de datos que se encuentren en diversos formatos y locaciones (de ser el caso), y centralizar todos en un almacén de datos, para posteriormente someterlo a diversos procesos que puedan extraer información relevante que nos otorgue un conocimiento determinado [19–21].

Cabe destacar que KDD no es algún *software* o programa que nos permita realizar el minado de datos, KDD es una metodología de múltiples pasos, algo así como el "Método Científico de la Ciencia de Datos", que nos ayudará a descubrir y explotar todo el potencial que podamos extraer de nuestros datos recabados.

Ésta metodología se centra particularmente en el "usuario" es decir, en no automatizar totalmente el proceso de extracción de información de los datos crudos, dado que el análisis humano en cada una de sus etapas es de gran importancia para la toma de decisiones de esta metodología, por lo cual termina siendo un proceso iterativo, donde las decisiones del usuario afectan el flujo del proceso para la correcta extracción de información; de lo contrario podríamos tener información totalmente inútil a nuestros objetivos, extraída por un proceso automatizado.

Por otro lado, también podemos emplear todo tipo de herramientas apropiadas para la visualización y análisis de los datos que nos permitan descubrir patrones en los mismos. Actualmente existe la tendencia a realizar el análisis automático de los datos a través de técnicas cada vez más sofisticadas de inteligencia artificial (IA), aunque la parte interpretativa humana sigue siendo hasta el momento irreemplazable.

La metodología KDD consta de cinco "Productos" (Datos iniciales, Almacén de datos, Datos seleccionados, Patrones y Conocimiento) y de cuatro "Procesos" ('Integración y recopilación', 'Selección, limpieza y transformación', 'Minería de datos' y 'Evaluación e interpretación'), así como se muestra en la Figura 4.1:

Figura 4.1: Diagrama de procesos KDD

A continuación, veremos detenidamente cada uno de estos procesos y los productos que se obtuvieron en esta investigación a partir del análisis del clima en el plantel San Lorenzo Tezonco de la UACM para los sitios marcados como puntos de monitoreo de la red de estaciones meteorológicas construidas exprofeso.

4.1. Datos Iniciales

Los datos iniciales son, "todos los datos o registros" de los cuales dispongamos para la investigación, son los llamados datos crudos y sin modificación alguna, ya sean archivos en papel que haya que digitalizar o archivos electrónicos como: CSV (archivos separados por comas), XLS (hojas de cálculo, Excel), TXT (texto plano), PDF, DOCX (Word), XML (estructuras de datos), Bases de Datos, Formularios, etc.

Serán todos estos los cuales integren el histórico dentro del Almacén de Datos. En nuestro caso son aquellos generados por las estaciones meteorológicas en formato TXT.

Las Estaciones Meteorológicas generan un promedio de 1,200 registros por hora o 28,800 registros al día cada una. Como disponemos de nueve estaciones meteorológicas, estas generan 10,800 registros por hora y 259,200 registros por un día de trabajo completo, poco más de un cuarto de millón de registros al día.

Cada archivo TXT generado por una estación meteorológica con 28,800 registros al día pesan 1,148 KB (Kilobyte). Recordemos que las estaciones cuentan con una memoria micro SD de 8 GB, lo cual implica que cada estación meteorológica tendrá una capacidad aproximada de trabajo de 167,247 horas de almacenamiento.

4.2. Integración, Recopilación y Almacén de Datos

En este proceso de la metodología KDD, se centra principalmente en reunir todos los "Datos Iniciales", archivos físicos y electrónicos pertinentes a la investigación, en nuestro caso los archivos generados por todas las estaciones meteorológicas. Es importante destacar que no importa el día, la fecha u hora en que se hayan generado o realizado las tomas de los datos, ya que estamos haciendo el concentrando de todos los archivos generados; a esto se le llamará el "Histórico", inclusive si uno o más archivos llegaron a presentar incongruencias o faltas de información por fallas en las estaciones, también serán tomados en cuenta. Todos los datos son importantes y no hay que descartar ninguno; posteriormente podremos hacer una selección más precisa de los archivos y registros que deseemos analizar.

Para esta etapa es recomendable generar una Base de Datos que contenga todos los registros capturados por las estaciones meteorológicas, concentrando de esta manera toda la información de los archivos generados en un solo lugar (Integración). Para esto haremos uso del administrador de bases de datos de *software* libre *PostgreSQL* versión 10 con su *software* de administración *pgAdmin* versión 4, bajo la licencia

de *PostgreSQL*. De esta manera podremos ir agregando los registros de cada archivo de cada estación meteorológica a la base de datos.

El Almacén de datos (o *Data Warehouse*, en inglés), es el producto resultante del proceso anterior, es decir el concentrado total de todos los datos históricos y relevantes para nuestra investigación. La principal función de nuestro almacén de datos es tener bien ubicados cada uno de nuestros archivos en un solo lugar, esto con el fin de no estar buscando posteriormente archivos faltantes.

En la actualidad, el Almacén de Datos es representado por una Base de Datos que integra toda la información relevante a la investigación (de ser necesario digitalizar archivos físicos, se hace), para que de esta manera la Base de Datos se pueda manejar posteriormente por diverso *software* para el análisis de los registros contenidos haciendo uso de las consultas SQL (*Structured Query Language*, por sus siglas en inglés) a la base de datos. Podemos ver una proyección (consulta) a nuestra base de datos en la Tabla 4.1, la cual irá integrando todos los registros de las estaciones meteorológicas.

	fecha date	hora time witho	estacion character	edificio character	posicion character	altura character	humedad numeric	tmp1 numeric	tmp2 numeric	presion numeric	polvo numeric	viento numeric
1	2017-09-02	12:00:01	e1	A	Y1	P0	40	21	22.5	78160	21.14	0
2	2017-09-02	12:00:04	e1	A	Y1	P0	40	21	22.5	78157	26.95	0
3	2017-09-02	12:00:07	e1	A	Y1	P0	40	21	22.6	78159	16.99	0
4	2017-09-02	12:00:10	e1	A	Y1	P0	40	21	22.6	78161	18.65	0
5	2017-09-02	12:00:13	e1	A	Y1	P0	40	21	22.6	78158	22.8	0
6	2017-09-02	12:00:16	e1	A	Y1	P0	40	21	22.6	78161	20.31	0
7	2017-09-02	12:00:19	e1	A	Y1	P0	40	21	22.6	78158	18.65	0
8	2017-09-02	12:00:22	e1	A	Y1	P0	40	21	22.6	78161	6.2	0
9	2017-09-02	12:00:25	e1	A	Y1	P0	40	21	22.6	78158	22.8	0
10	2017-09-02	12:00:29	e1	A	Y1	P0	40	21	22.6	78159	20.31	0
11	2017-09-02	12:00:32	e1	A	Y1	P0	40	22	22.6	78159	23.63	0
12	2017-09-02	12:00:35	e1	A	Y1	P0	40	22	22.6	78168	21.97	0
13	2017-09-02	12:00:38	e1	A	Y1	P0	40	22	22.6	78163	16.99	0

Tabla 4.1: Proyección de la base de datos

Una vez recopilados todos los archivos (registros), es importante tener siempre uno o más respaldos de nuestro Almacén de Datos y en distintas locaciones para evitar la pérdida total o parcial de estos, previendo cualquier inconveniente futuro que pudiera resultar de un error humano, desastre natural y/o fallas en las computadoras o servidores donde se están almacenan los archivos. Para ver como se creó la base de datos en *PostgreSQL* puede revisar el Apéndice C.1. Nota: Cuando los datos son relativamente pocos (decenas o centenas de millones de registros) se pueden respaldar en discos externos o cintas, pero cuando llegan a ser del orden de *Terabytes* o *Petabytes* por día deben respaldarse en otra infraestructura.

4.3. Selección, Limpieza, Transformación y Datos Seleccionados

En este proceso de la metodología, se llevará a cabo la selección de los archivos que deseemos analizar, una vez que hemos concentrado toda la información, podemos empezar a seleccionar secciones y/o partes del material para hacer diversos análisis de minería, ya sea por periodos de tiempo, categorías, variables, rangos, niveles, etc. de tal manera que en los procesos sucesivos no se trabaje de más extrayendo información no relevante a los objetivos planteados y/o con las reglas del negocio.

A este proceso también le podemos llamar "Selección de vistas minables", el cual consistirá en la creación de archivos con datos específicos con los cuales trabajar posteriormente.

Para nuestro primer análisis seleccionaremos una muestra de 24 horas, de las nueve estaciones meteorológicas, empezando a las 00:00:00 horas y terminando a las 23:59:59 horas, además restringiendo que la altura de todas las estaciones sea a nivel de piso (Planta Baja). De esta manera podemos ir creando nuevas vistas minables conforme avanzamos en el proceso de extracción de información.

Una vez seleccionado los datos podemos hacer una primera limpieza de los mismos, en este caso haremos un promedio de los registros por minuto, reduciendo de forma considerable los registro para su cálculo

de 28800 a 1440, conservando la integridad de la información. Podemos ver como quedan ahora los registros en la Tabla 4.2. Esto lo podemos hacer desde nuestra base de datos de la siguiente manera (ver Apéndice C.2).

Nota: Los promedios por minuto se manejan para disminuir el análisis final de los datos y por qué las variaciones cada tres segundos son muy regulares sin cambios de orden en las mediciones.

	fecha date	hora text	estacion character	edificio character	posicion character	altura character	humedad numeric	tmp1 numeric	tmp2 numeric	presion numeric	polvo numeric	viento numeric
11	2017-09-02	12:10	e1	A	Y1	P0	39.00	22.00	23.55	78159.05	23.98	0.00
12	2017-09-02	12:11	e1	A	Y1	P0	38.11	22.89	23.65	78152.68	22.54	0.00
13	2017-09-02	12:12	e1	A	Y1	P0	38.00	23.00	23.83	78150.95	21.68	0.00
14	2017-09-02	12:13	e1	A	Y1	P0	38.00	23.00	23.88	78151.74	20.70	0.00
15	2017-09-02	12:14	e1	A	Y1	P0	38.00	23.00	23.89	78151.63	24.55	0.00
16	2017-09-02	12:15	e1	A	Y1	P0	38.00	23.00	23.93	78152.85	20.73	0.00
17	2017-09-02	12:16	e1	A	Y1	P0	38.00	23.00	24.00	78154.37	19.35	0.00
18	2017-09-02	12:17	e1	A	Y1	P0	38.00	23.00	24.09	78152.37	24.02	0.00
19	2017-09-02	12:18	e1	A	Y1	P0	38.00	23.00	24.20	78155.35	21.80	0.00
20	2017-09-02	12:19	e1	A	Y1	P0	38.00	23.00	24.18	78153.74	28.05	0.00
21	2017-09-02	12:20	e1	A	Y1	P0	37.95	23.00	24.20	78150.26	20.22	0.00
22	2017-09-02	12:21	e1	A	Y1	P0	37.05	23.00	24.19	78148.47	24.11	0.00
23	2017-09-02	12:22	e1	A	Y1	P0	37.00	23.00	24.11	78150.50	23.63	0.00

Tabla 4.2: Proyección de la base de datos (registros por minuto)

Finalmente para terminar esta etapa, crearemos cinco vistas minables más transformando nuestra base de datos para obtener primero nueve tablas (una por estación meteorológica con los promedios por minutos), y luego una por cada tipo de variable específica (presión, temperatura, humedad, densidad de polvo y velocidad del viento), cada tabla contendrá exclusivamente la hora de los registros y las nueve lecturas de cada una de las estaciones por tipo de variable determinada, recordemos que ya seleccionamos previamente los registros de un día en específico. Para esto se puede consultar el Apéndice C.3 realizando las instrucciones desde nuestra base de datos.

Una vez que tenemos nuestras tablas podemos extraer en forma de archivos nuestros *Datos Seleccionados* con las cuales trabajar; éstos estarán preparados para aplicarles funciones y algoritmos más especializados que nos entreguen la información que estamos buscando para cumplir nuestros objetivos.

Nuestras vistas minables constan de cinco archivos o tablas por tipo de variable atmosférica:

- Humedad
- Temperatura
- Presión
- Densidad de Polvo
- Velocidad del Viento

4.4. Minería de Datos y Patrones

Para la implementación de la minería de datos haremos uso del lenguaje de programación R [22, 23], y el entorno de desarrollo RStudio [24] los cuales se distribuyen bajo licencia de *software* libre GNU GPL (*General Public License*) y GNU AGPL (*Affero General Public License*) respectivamente. R es un lenguaje popular para la computación estadística, proporcionando una amplia variedad de técnicas y modelado (lineal y no lineal), pruebas clásicas de estadística, clasificaciones, agrupaciones y análisis de series de tiempo entre otras técnicas, así como distintas aplicaciones para gráficos de calidad.

Para empezar con nuestro análisis de datos, es necesario definir qué queremos saber o descubrir de los mismos. Recordemos que el principal objetivo de recolectar datos con las estaciones meteorológicas, es describir cómo se comporta el clima dentro del plantel San Lorenzo Tezonco y si es el caso inferir si hay patrones asociados a condiciones locales o efectos de mesoescala. Teniendo esto en cuenta hemos creado las vistas minables en el proceso anterior por lo cual ahora podremos trabajar con ellas, empezando por ver como es el comportamiento de cada una de nuestras variables ambientales en el transcurso del día (viento, humedad, polvo, presión barométrica y temperatura) y terminando por analizar cuál es su comportamiento dentro de los tres edificios principales del plantel (Edificio A, B y C), como se muestra en la Figura 4.2.

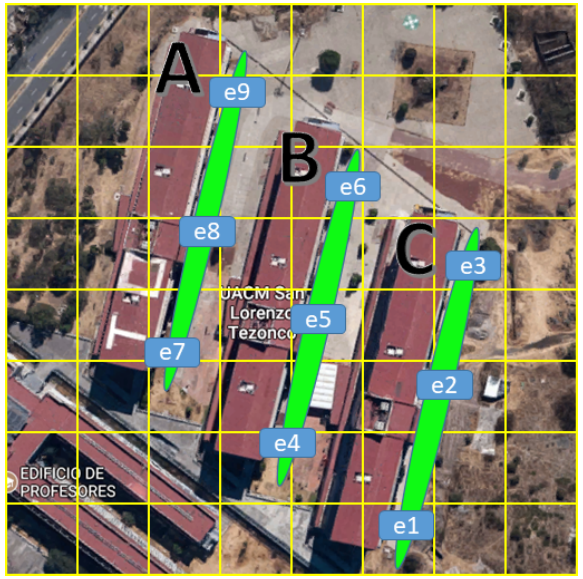

Figura 4.2: Edificios principales de clases

Haciendo uso de R podemos extraer los "patrones" que revelen información que a simple vista, al ver los registros de forma numérica, no podamos percibir; esto se logra implementando de forma gráfica los resultados obtenidos al aplicar diversos algoritmos y cálculos a nuestros datos seleccionados.

Para empezar, analizamos de forma individual cada variable ambiental durante el transcurso del día, haciendo uso de la librería "qcc" para R, la cual está diseñada para gráficos de control de calidad estadístico (*quality control charts*), fue creada por Luca Scrucca con su última versión 2.7 escrita en 2017-07-09. Para ver la implementación en R ver el Apéndice D.1.

Nota: Haciendo uso de la metodología KDD, primeramente mostraremos los gráficos de las estaciones, ya que estamos en el proceso de Minería de Datos, del cual extraemos los Patrones (gráficos y estadísticos), para posteriormente en el siguiente proceso, Evaluación e Interpretación, poder analizarlos.

En la Figura 4.3 podemos apreciar el comportamiento de las variables de Humedad de cada una de las estaciones con los siguientes estadísticos:

e1	e2	e3	e4	e5
Min. :33.50	Min. :33.00	Min. :32.00	Min. :34.00	Min. :31.00
1st Qu.:36.00	1st Qu.:36.00	1st Qu.:34.00	1st Qu.:36.00	1st Qu.:32.00
Median :38.00	Median :38.00	Median :37.92	Median :38.00	Median :35.00
Mean :37.84	Mean :37.82	Mean :37.04	Mean :37.85	Mean :34.49
3rd Qu.:40.36	3rd Qu.:40.00	3rd Qu.:40.00	3rd Qu.:40.00	3rd Qu.:37.00
Max. :42.00	Max. :42.00	Max. :42.00	Max. :42.00	Max. :38.00

e6	e7	e8	e9
Min. :31.50	Min. :31.00	Min. :34.00	Min. :32.75
1st Qu.:33.37	1st Qu.:32.04	1st Qu.:36.00	1st Qu.:34.69
Median :36.00	Median :34.65	Median :38.00	Median :37.00
Mean :35.77	Mean :34.57	Mean :37.61	Mean :36.69
3rd Qu.:38.50	3rd Qu.:37.00	3rd Qu.:39.00	3rd Qu.:38.75
Max. :40.00	Max. :38.00	Max. :41.00	Max. :40.50

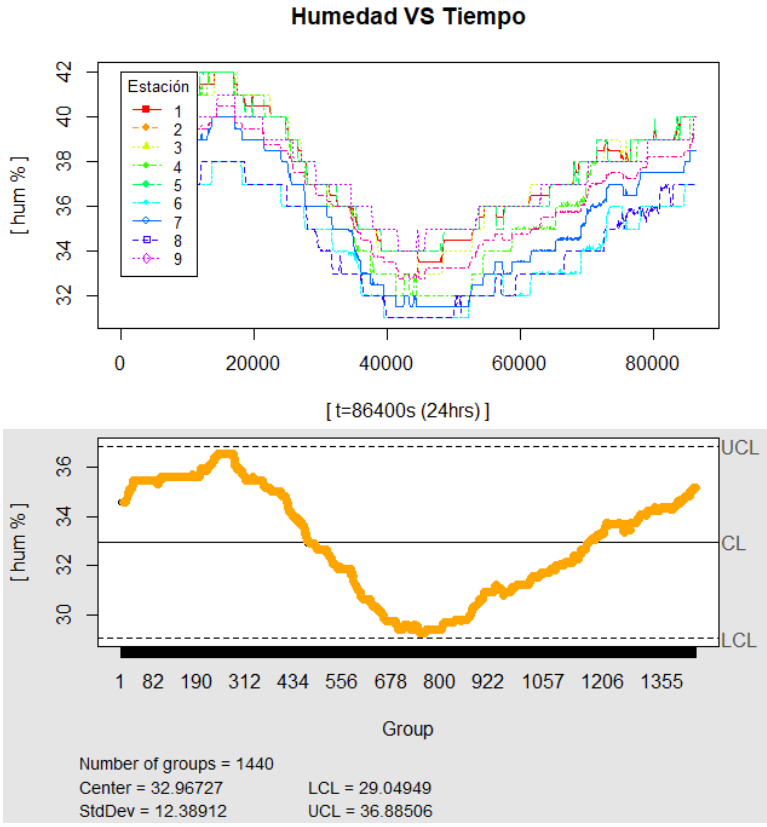

Figura 4.3: Humedad relativa porcentual

En la Figura 4.4 podemos apreciar el comportamiento de las variables de Temperatura de cada una de las estaciones con los siguientes estadísticos:

e1	e2	e3	e4	e5
Min. :10.05	Min. :10.70	Min. : 7.80	Min. : 9.40	Min. : 9.14
1st Qu.:12.77	1st Qu.:13.29	1st Qu.:10.61	1st Qu.:12.20	1st Qu.:11.90
Median :16.29	Median :16.91	Median :14.50	Median :15.72	Median :14.98
Mean :17.01	Mean :17.61	Mean :15.41	Mean :16.41	Mean :15.57
3rd Qu.:20.60	3rd Qu.:20.90	3rd Qu.:19.50	3rd Qu.:20.14	3rd Qu.:18.96
Max. :24.85	Max. :26.09	Max. :25.02	Max. :23.70	Max. :22.10

e6	e7	e8	e9
Min. : 8.55	Min. : 9.80	Min. : 9.80	Min. : 9.20
1st Qu.:11.34	1st Qu.:12.50	1st Qu.:12.60	1st Qu.:11.97
Median :14.74	Median :15.79	Median :15.78	Median :15.25
Mean :15.49	Mean :16.31	Mean :16.29	Mean :15.89
3rd Qu.:19.23	3rd Qu.:19.80	3rd Qu.:19.70	3rd Qu.:19.46
Max. :23.40	Max. :22.82	Max. :22.60	Max. :22.81

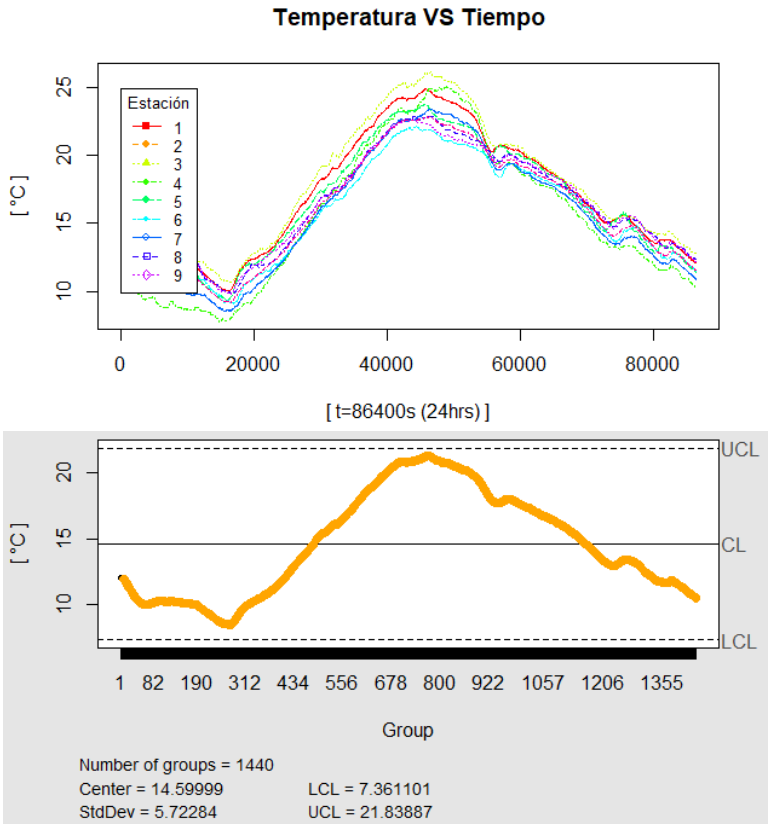

Figura 4.4: Temperatura ambiental en °C

En la Figura 4.5 podemos apreciar el comportamiento de las variables de Presión de cada una de las estaciones con los siguientes estadísticos:

```
       e1              e2              e3              e4              e5
 Min.   :77825   Min.   :77813   Min.   :77937   Min.   :77835   Min.   :78300
 1st Qu.:77985   1st Qu.:77976   1st Qu.:78114   1st Qu.:77995   1st Qu.:78458
 Median :78115   Median :78105   Median :78252   Median :78125   Median :78594
 Mean   :78096   Mean   :78087   Mean   :78225   Mean   :78105   Mean   :78572
 3rd Qu.:78226   3rd Qu.:78218   3rd Qu.:78355   3rd Qu.:78235   3rd Qu.:78697
 Max.   :78309   Max.   :78302   Max.   :78437   Max.   :78319   Max.   :78781
       e6              e7              e8              e9
 Min.   :78122   Min.   :77801   Min.   :77774   Min.   :77950
 1st Qu.:78282   1st Qu.:77954   1st Qu.:77941   1st Qu.:78112
 Median :78422   Median :78096   Median :78081   Median :78252
 Mean   :78399   Mean   :78073   Mean   :78063   Mean   :78231
 3rd Qu.:78526   3rd Qu.:78203   3rd Qu.:78199   3rd Qu.:78363
 Max.   :78605   Max.   :78284   Max.   :78286   Max.   :78445
```

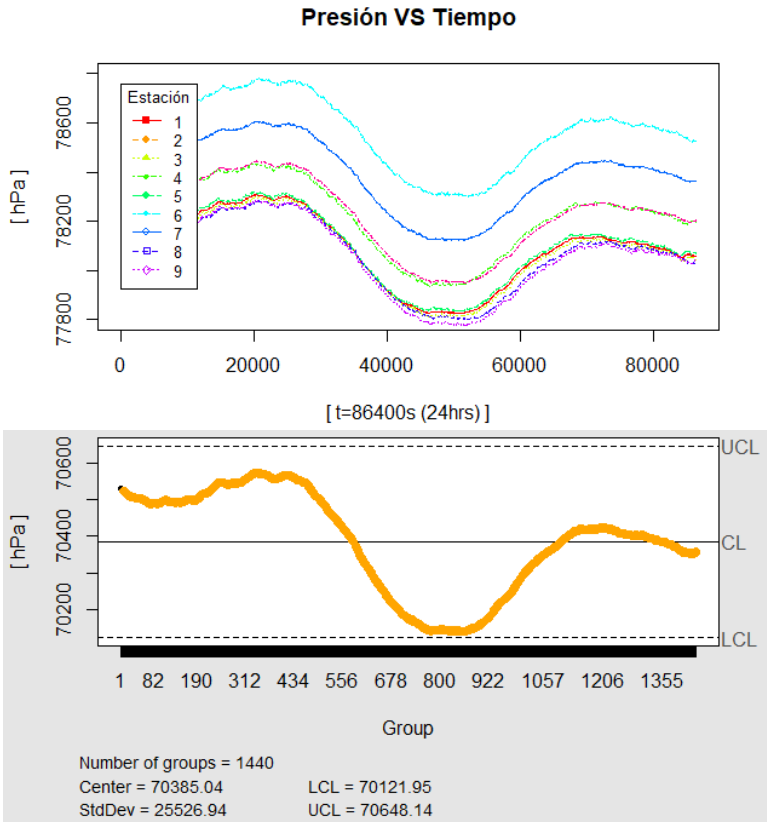

Figura 4.5: Presión barométrica en hectopascales

En la Figura 4.6 podemos apreciar el comportamiento de las variables de Polvo de cada una de las estaciones con los siguientes estadísticos:

```
       e1                e2                e3                e4                e5
 Min.   :0.01000   Min.   :0.000000   Min.   :0.03000   Min.   :0.00000   Min.   :0.1900
 1st Qu.:0.01000   1st Qu.:0.010000   1st Qu.:0.04000   1st Qu.:0.01000   1st Qu.:0.2000
 Median :0.02000   Median :0.010000   Median :0.04000   Median :0.02000   Median :0.2100
 Mean   :0.01544   Mean   :0.008757   Mean   :0.04113   Mean   :0.01792   Mean   :0.2112
 3rd Qu.:0.02000   3rd Qu.:0.010000   3rd Qu.:0.04000   3rd Qu.:0.02000   3rd Qu.:0.2200
 Max.   :0.04000   Max.   :0.040000   Max.   :0.06000   Max.   :0.06000   Max.   :0.2300
       e6                e7                e8                e9
 Min.   :0.1200   Min.   :0.05000   Min.   :0.1100   Min.   :0.110
 1st Qu.:0.1200   1st Qu.:0.06000   1st Qu.:0.1200   1st Qu.:0.120
 Median :0.1300   Median :0.07000   Median :0.1300   Median :0.130
 Mean   :0.1283   Mean   :0.07166   Mean   :0.1263   Mean   :0.128
 3rd Qu.:0.1400   3rd Qu.:0.09000   3rd Qu.:0.1400   3rd Qu.:0.140
 Max.   :0.1400   Max.   :0.10000   Max.   :0.1500   Max.   :0.150
```

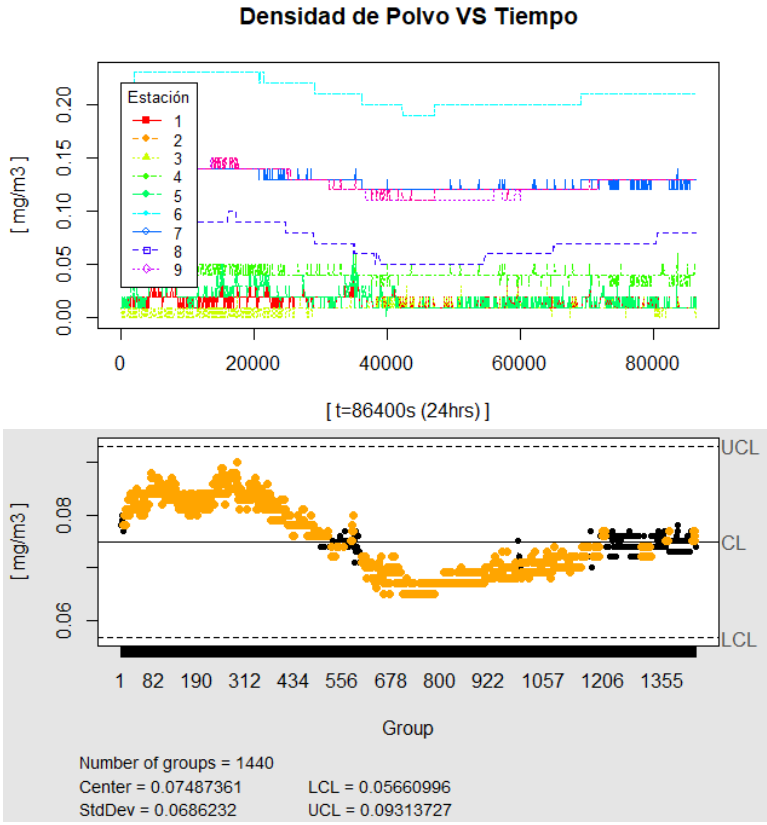

Figura 4.6: Densidad de polvo en el aire en mg/m3

En la Figura 4.7 podemos apreciar el comportamiento de las variables de Viento de cada una de las estaciones con los siguientes estadísticos:

e1		e2		e3		e4		e5	
Min.	:0.000000	Min.	:0	Min.	:0	Min.	:0.000000	Min.	:0
1st Qu.	:0.000000	1st Qu.	:0	1st Qu.	:0	1st Qu.	:0.000000	1st Qu.	:0
Median	:0.000000	Median	:0	Median	:0	Median	:0.000000	Median	:0
Mean	:0.001167	Mean	:0	Mean	:0	Mean	:0.002326	Mean	:0
3rd Qu.	:0.000000	3rd Qu.	:0	3rd Qu.	:0	3rd Qu.	:0.000000	3rd Qu.	:0
Max.	:1.630000	Max.	:0	Max.	:0	Max.	:3.250000	Max.	:0

e6		e7		e8		e9	
Min.	:0	Min.	:0.000000	Min.	:0.000000	Min.	:0.000000
1st Qu.	:0	1st Qu.	:0.000000	1st Qu.	:0.000000	1st Qu.	:0.000000
Median	:0	Median	:0.000000	Median	:0.000000	Median	:0.000000
Mean	:0	Mean	:0.002847	Mean	:0.008333	Mean	:0.004188
3rd Qu.	:0	3rd Qu.	:0.000000	3rd Qu.	:0.000000	3rd Qu.	:0.000000
Max.	:0	Max.	:1.800000	Max.	:3.800000	Max.	:1.900000

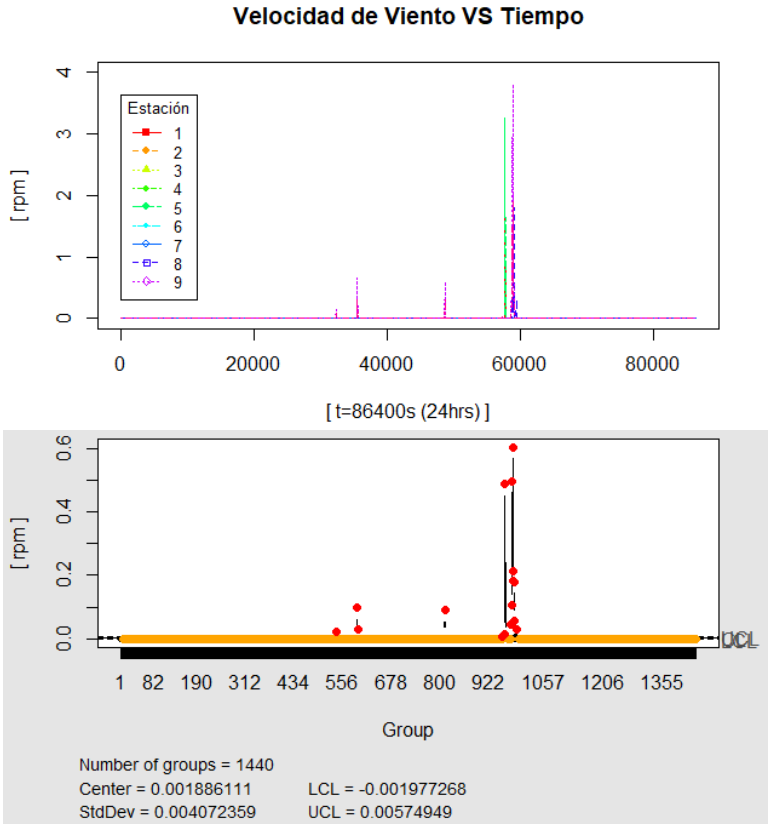

Figura 4.7: Rapidez del viento en rpm del anemómetro

Una vez que hemos podido ver el comportamiento de las variables de forma visible e independiente, podemos empezar a preguntarnos qué significan y cómo es que se comportan de forma espacial, es decir en un plano (distribuidas dentro del plantel). Para esto haremos uso del método de interpolación por *Kriging*, el cual es un método geoestadístico para la aproximación de puntos [25].

Para implementar las funciones de *Kriging* en R, tenemos que importar la librería "geoR" la cual fue desarrollada en 2016 para el análisis de datos geoestadísticos por Paulo J. Ribeiro Jr. [26, 27] Cabe mencionar que para este tipo de análisis geoestadísticos existen diversas librerías que se pueden implementar, inclusive programar nuestros propios algoritmos desde cero, si ése es el caso.

Para poder implementar los algoritmos de *Kriging*, necesitamos previamente dos cosas, correspondientes a un tiempo determinado:

- La distribución espacial de las estaciones meteorológicas y
- El vector de datos.

La configuración espacial de las estaciones meteorológicas e1, e2, . . . , e9, se muestra en la Figura 4.8. Nótese la adaptación en paralelo de los edificios, esto con el fin de mejorar (adaptar) el modelo matemático de coordenadas X e Y:

Edificio A: e7, e8, e9 - o de forma vectorial: (0,2), (1,2), (2,2)
Edificio B: e4, e5, e6 - o de forma vectorial: (0,1), (1,1), (2,1)
Edificio C: e1, e2, e3 - o de forma vectorial: (0,0), (1,0), (2,0)

Los vectores de datos que tomaremos para el análisis serán: temperatura, presión atmosférica, humedad porcentual, densidad de polvo y velocidad del viento. Cada uno de estos vectores consta de nueve valores, uno por cada estación en su posición determinada.

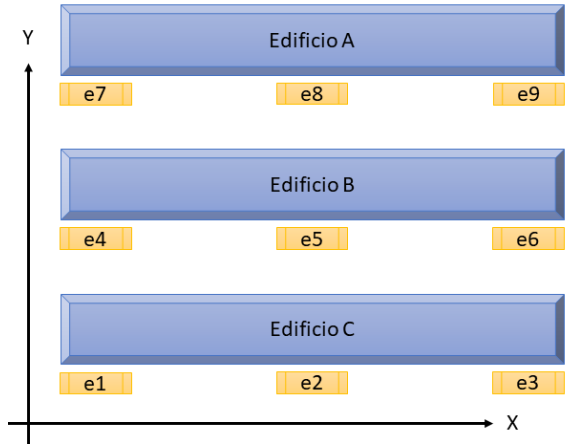

Figura 4.8: Configuración espacial de las EM

De forma general, nos interesa saber cómo podrían afectar las condiciones climatológicas en el plantel a la comunidad estudiantil. Por esta razón se deciden tomar los valores de las horas pico de tránsito de personas, siendo ésta a las 14:30 horas, que es donde se estima el flujo máximo de personas que entran (comunidad estudiantil de la tarde) y salen (comunidad estudiantil de la mañana).

Analizando estas variables con el método de interpolación por *Kriging* con media conocida y una sub-malla de 200x200 puntos en R obtenemos los siguientes gráficos o "patrones" (Figuras 4.9, 4.10, 4.11 y 4.12). Para poder ver como se implementó el método de interpolación por Kriging en R para obtenerlos siguientes patrones ver el Apéndice D.2.

Nota: Las lecturas para el anemómetro (velocidad del viento) fueron mínimas, por lo cual no se pudo generar el modelo matemático, dado que las lecturas para ese día fueron prácticamente cero, lo que implica que ese día no hubo mucho viento en el plantel. Puede observar las lecturas en la Figura 23. Además, tómese en cuenta que el dispositivo para medir la velocidad del viento no es muy preciso dado que la hélice es muy pesada para variaciones bajas de aire y es pesado para vencer la inercia del sensor (aspas).

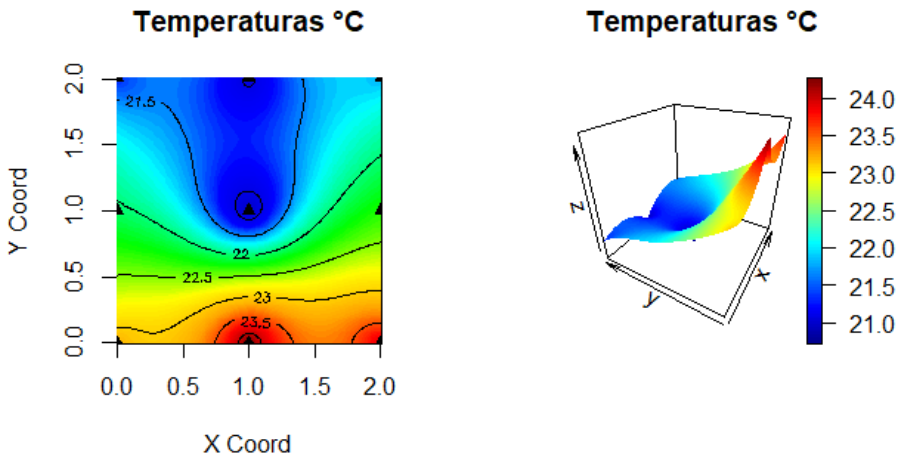

Figura 4.9: Kriging de Temperaturas en °C

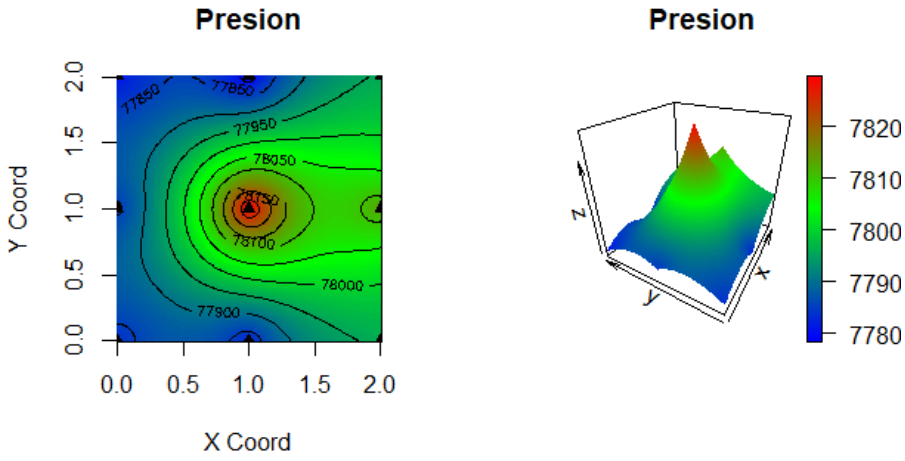

Figura 4.10: Kriging de Presiones en hPa

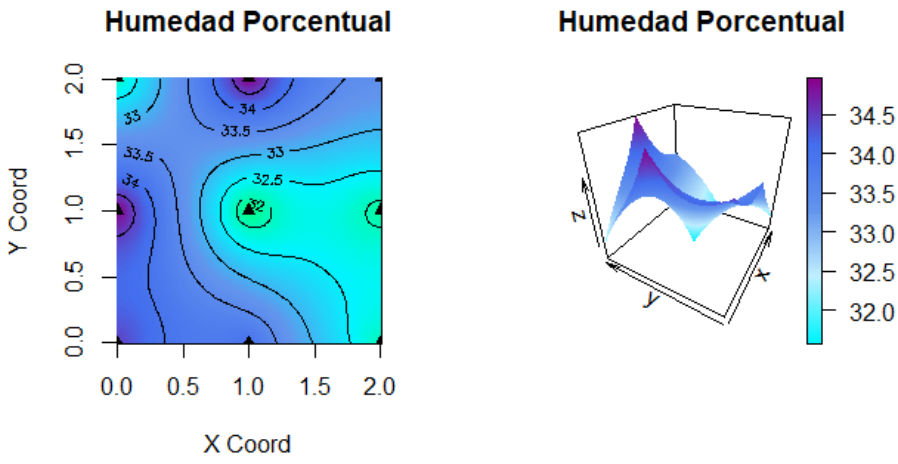

Figura 4.11: Kriging de humedad porcentual

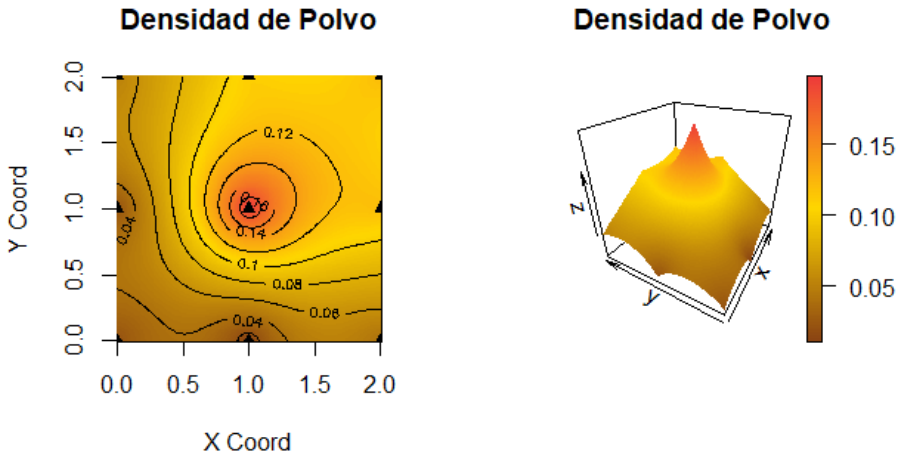

Figura 4.12: Kriging de densidad de polvo en mg/m3

4.5. Evaluación, Interpretación y Conocimiento

Finalmente llegamos al proceso de "Evaluación e Interpretación" para la extracción de información que nos brinde un "Conocimiento" determinado sobre las variables de estudio. Para ello analizaremos cada uno de nuestros patrones obtenidos por el método de *Kriging* y del comportamiento individual de cada una de las variables.

Para empezar tenemos la disposición gráfica del comportamiento de las variables por cada una de las estaciones donde R nos proporciona los estadísticos básicos mediante la función *summary()* y de las formas gráficas con las funciones *plot()* y *lines()* que nos ayudarán a construirlos.

Podemos apreciar en las Figuras 4.3, 4.4, 4.5, 4.6 y 4.7 el comportamiento de las variables en cada una de las estaciones meteorológicas, en las cuales se aprecia el comportamiento similar entre cada una de ellas y sus variables respectivas difiriendo solo en magnitud. Aquí podríamos apreciar si es que una de los sensores de las estaciones podría estar presentando fallos en su funcionamiento al estar registrando valores con un comportamiento diferente a las demás.

Para las variables de humedad, temperatura y presión (Figuras 4.3, 4.4 y 4.5 respectivamente), Podemos apreciar un comportamiento correlacional entre las variables lo cual nos ayuda a identificar exactamente esta relación entre variables para poder generar posteriormente en una nueva iteración de la metodología KDD para poder analizar estas variables en correlaciones mutuas.

Por otro lado, tenemos las variables de densidad de polvo y de rapidez de viento. Como se puede apreciar en sus gráficas (figuras 4.6 y 4.7 respectivamente), estas no están tan directamente relacionadas entre ellas dado que los factores de partículas de polvo en el aire dependen de otros factores como por ejemplo la cantidad de carros que circulan en ese día. Es de esperar que la velocidad del viento sea afectada por los cambios de presión y temperatura, pero como se había mencionado antes, la sensibilidad del sensor no es apto para detectar corrientes de aire muy bajas.

Para terminar con esta primera parte es importante mencionar la distribución estadística de las lecturas respecto a su media, mediana, varianza y controles de acotamiento (UCL y LCL, del inglés *Upper Control Limit and Lower Control Limit* [28]) las cuales muestran un control de acotamiento en n-sigmas (donde 1 sigma es equivalente a la varianza). Por *default* esta última se utiliza, en la función *qcc()* para graficar de la librería con el mismo nombre para R, con un valor de 3 sigma la cual determina que el conjunto total de datos registrados cae a una distancia no mayor a tres del valor de la mediana o centro de la distribución de los datos. Para las gráficas de las variables (Figuras 4.3, 4.4, 4.5, 4.6 y 4.7), se emplearon las siguientes sigmas:

- Humedad: sigma=1
- Temperatura: sigma=4
- Presión atmosférica: sigma=0.026
- Densidad de polvo: sigma=0.6
- Velocidad de viento: sigma=3

Como segunda parte de análisis, tenemos las figuras 4.9, 4.10, 4.11 y 4.12 de las variables temperatura, presión, humedad y densidad de polvo respectivamente, que a simple vista podemos notar en conjunto como existe un "encajonamiento" de nuestras variables derivado de la disposición de los edificios en el plantel. Recordemos que hemos seleccionado las estaciones a nivel de piso (o planta baja) para las 14:30 horas.

Para las figuras 4.4 y 4.9 tenemos la variable de temperatura, de la cual podemos observar cómo es que la temperatura gracias a la posición del sol en dirección Sur, impacta más al edificio C lo cual provoca que proyecte su sobra al edificio B y él a su vez al edificio A, lo cual reduce considerablemente la temperatura en el edificio A y B, siendo estos los que reciben menos luz solar.

Para las figuras 4.6 y 4.12 tenemos la variable de polvo, de la cual observamos como existe una mayor concentración del mismo en dirección Norte-Oeste, en la cual se encuentra el cerro de la mina de San

Lorenzo Tezonco, teniendo de igual manera una alta concentración en medio del edificio B derivado del efecto de encajonamiento.

Y finalmente para las variables de presión (figuras 4.5 y 4.10) y de humedad (figuras 4.3 y 4.11) podemos observar que tiene una relación por la disposición de sus lecturas sobre los edificios, en particular del edificio B, donde hay una menor humedad en dirección Sur-Oeste y la presión tiende a subir; ambos con sus lecturas máximas y mínimas en centro del edificio B.

Capítulo 5

Conclusiones

En el transcurso de toda la investigación que comenzó simplemente por saber ¿cómo es que se comporta el clima dentro de la universidad?, se destacaron como puntos más importantes lo siguiente:

Parte I
Sobre los datos analizados

1. El proyecto está en sus comienzos y por esta razón solo tuvimos nueve estaciones meteorológicas para hacer la medición de datos, por esta razón es que se hace estadística de un día a lo más.

2. Se tomaron mediciones para tres niveles y se guardaron en un *cluster* que hace todo el análisis de datos.

3. Debido a la temporalidad de las mediciones de las nueve estaciones (un día de mediciones) la distribución de las variables aleatorias medidas (Presión, Temperatura, etc.) no es normal, se tomaron promedios por minuto y se obtuvieron los principales estadísticos.

4. Las gráficas superpuestas de cada variable por cada estación tienen el mismo comportamiento cualitativo, aunque no se hizo la correlación de variables podría encontrarse por ejemplo entre presión y temperatura.

5. Las curvas de nivel con los datos dispersos fueron hechas por un método matemático llamado *Kriging* y suaviza las curvas encontrando datos interpolados. Las curvas de nivel como isobaras

(igual presión) muestran las zonas de mínima presión y pueden deberse a ciertas islas provocadas por la forma larga de los edificios

6. Para hacer un análisis con más días de muestreo y más estaciones meteorológicas, parte del trabajo futuro; se necesitarán implementar más algoritmos distribuidos y de *Big Data*, lo cual se seguirá trabajando con el *software* R que puede implementar dichos algoritmos.

Parte II
Sobre la tecnología usada

1. Con las estaciones construidas se hicieron mediciones manejadas con la infraestructura de cómputo disponible. Para un futuro se planea la necesidad de construir una red de estaciones meteorológicas más amplia (del orden de 500-1000). Esto con la finalidad de cubrir una mayor área del plantel y poder hacer mejores modelaciones matemáticas de las variables atmosféricas, que se reflejaran en mejores interpolaciones de nuestra malla de estaciones disponibles.

2. El tiempo de operación de las estaciones depende de la carga y batería usadas que es de unas 48 horas de autonomía en promedio, dado que se distribuyen lejos de contactos eléctricos para medir en puntos representativos. Se plantea la necesidad de hacer más "autónomas" cada una de las estaciones meteorológicas, derivado de las siguientes razones:

   - Dado que por poseer pilas recargables las mismas no pueden estar funcionando por largos periodos de tiempo.
   - Al contar con un sistema de almacenamiento por memoria micro SD, dispone de una capacidad limitada de almacenamiento.

3. La autonomía que se busca de las estaciones es que en una siguiente etapa de la investigación se puedan actualizar las Estaciones Meteorológicas, las placas *Arduino* y/o a *Rasberry* con el fin de cubrir los siguientes objetivos:

- Implementar una proteccion para el *hardware* que resista las condiciones climatológicas del ambiente al que están expuestas.
- Adaptar más o mejores sensores en placas Arduino más recientes u otro *hardware* con mayor capacidad de puertos.
- Implementar un sistema de transmisión de datos inalámbricos usando tecnología *Wi-Fi* a un servidor construido con *Rasberry*, entre cada cierto número de estaciones en la red.

4. Se construyó un *cluster* con equipos Dell con procesadores *Pentium* IV HT con una red *Gigabit*, etc. que pueden ser considerados de uso y desecho, pero ideal para el aprendizaje. En un futuro se planea la necesidad de un *cluster* más poderoso, ampliando el número de nodos en la red tipo *Xeon*, o inclusive con nodos instalados en *racks*. Esto para la implementación y aplicación de mejores algoritmos en paralelo y distribuidos que puedan tener como meta, ampliar la capacidad de cálculo y obtener todos los resultados en un tiempo mucho menor.

Apéndice A

Manual detallado de configuración e implementación de un *Cluster Beowulf* con CentOS 7

En el presente apéndice se verá paso a paso la configuración que se llevó en el *Cluster Beowulf* dentro de cada uno de los nodos, maestro y esclavos, haciendo uso de la terminal.

A.1. Configuración general previa

1. Instalar Centos 7 en el nodo Maestro, con interfaz gráfica
2. Instalar Centos 7 (mínima) en los Nodos de Cálculo, sin interfaz gráfica.
3. Password homólogo root: X-X-X-X
4. Actualizar despues de la INSTALACIÓN
   a) yum -y update
5. Instalar repositorios:

a) yum install -y epel-release

b) yum install -y net-tools

c) yum install -y rsync

d) yum install -y nfs-utils

(CONFIGURACIÓN de cable RJ45, de izquierda a derecha)

1. blanco naranja
2. naranja
3. blanco verde
4. azul
5. blanco azul
6. verde
7. blanco café
8. café

A.2. Creación de nodo maestro

1. Deshabilitar NetworkManager de Centos 7
2. CONFIGURACIÓN de Redes: /etc/sysconfig/network-scripts/

   a) Red Externa:

      1) IPADDR
      2) NETMASK
      3) GATEWAY
      4) DNS
      5) NETWORK
      6) BROADCAST

   b) Red Interna:

1) IPADDR
2) NETMASK
3) GATEWAY

c) En cada archivo de CONFIGURACIÓN de interfaces de red: ifcfg-enp0s0

1) Agregar línea al final del archivo:
a′ NM_CONTROLLED=no

3. Deshabilitar y habilitar NetworkManager de Centos 7

a) systemctl stop NetworkManager
b) systemctl disable NetworkManager
c) systemctl start NetworkManager
d) systemctl enable NetworkManager
e) (NOTA: hacer lo mismo para el servicio network.service)

4. Deshabilitar Selinux

a) En el archivo de CONFIGURACIÓN del servicio selinux: /etc/sysconfig/selinux

Editar línea donde esté: SELINUX=enforcing a SELINUX=disabled

1. CONFIGURACIÓN hostname y dirección IP

a) En el archivo de CONFIGURACIÓN: /etc/hosts

1) Agregar las líneas:

10.0.0.1 master.beowulf_uacm.mx master
10.0.xy.z nodoxyz.beowulf_uacm.mx nodoxyz
... etc para cada nodo adicional del cluster

1. *a*) En el archivo de CONFIGURACIÓN que contiene el nombre de la máquina y opciones de red del sistema: /etc/sysconfig/network

1) Agregar líneas:

NOZEROCONF=yes HOSTNAME=master.beowulf_uacm.mx (ó simplemente master) NETWORKING=yes

1. *a*) Reiniciar el nodo Maestro.

1) reboot

A.3. Creación de nodo de cálculo

1. Deshabilitar y habilitar NetworkManager de Centos 7
    a) systemctl stop NetworkManager
    b) ...
2. En cada archivo de CONFIGURACIÓN de interfaces de red: ifcfg-enp0s0
    a) Agregar línea al final del archivo:
        1) NM_CONTROLLED=no
3. Deshabilitar Selinux
    a) En el archivo de CONFIGURACIÓN del servicio selinux: /etc/sysconfig/selinux
        1) Editar línea donde esté:

SELINUX=enforcing a SELINUX=disabled

1. CONFIGURACIÓN de direcciones IPs
    a) Copiar el archivo, /etc/hosts (del nodo maestro), a los nodos xyz
        1) scp root@master:/etc/hosts /etc/
    b) Verificar que la IP del nodo esté agregada en el archivo
        1) 10.0.xy.z nodoxyz.beowulf_uacm.mx nodoxyz
2. CONFIGURACIÓN hostname: En el archivo de CONFIGURACIÓN que contiene el nombre de la máquina y opciones de red del sistema: /etc/sysconfig/network
    a) Agregar las líneas:

NOZEROCONF=yes
HOSTNAME=peonXYZ.beowulf_uacm.mx
NETWORKING=yes

1. Reiniciar el nodo de cálculo
    a) reboot
2. (NOTA: Hacer esto para cada nodo de cálculo)

A.4. Instalación Ganglia (nodo maestro)

1. NOTA: Checar el tutorial: "Install and Configure Ganglia Monitoring System on CentOS 7"
   a) https://www.youtube.com/watch?v=tjHBYp649Ps
2. Navegar a la página web: http://ganglia.info
3. Bajar Ganglia Monitor core versión 3.7.2, el archivo se llama:
   a) ganglia-3.7.2.tar.gz
4. Instalar dependencias para construir el paquete rpm
   a) yum install -y freetype-devel rpm-build php httpd libpng-devel libart_lgpl-devel python-devel pcre-devel autoconf automake libtool expat-devel rrdtool-devel apr-devel gcc-c++ make pkgconfig
   b) yum install -y libconfuse-devel
5. Construir paquete rpm (en el directorio donde se descargó el tar.gz)
   a) rpmbuild -tb ganglia-3.7.2.tar.gz
6. Instalar Ganglia Monitor Core
   a) cd /root/rpmbuild/RPMS/x86_64/
   b) yum install -y *.rpm (todo lo de ganglia)
7. Instalar Web Server
   a) yum install httpd php php-mysql php-gd php-ldap php-odbc php-pear php-xml php-xmlrpc php-mbstring php-snmp php-soap curl curl-devel
8. Instalar Ganglia Server

a) yum install rrdtool rrdtool-devel ganglia-web ganglia-gmetad ganglia-gmond ganglia-gmond-python httpd apr-devel zlib-devel libconfuse-devel expat-devel pcre-devel

9. Verificar estén las IP's de todos los nodos y del maestro en /etc/hosts
10. Deshabilitar Firewall y el Selinux

A.5. Instalación Ganglia (nodo de cálculo)

1. Sincronizar archivos rpm ganglia del nodo maestro con nodo de cálculo:

   a) Crear directorio destino en el nodo de cálculo:
      1) cd (nota: en la raiz)
      2) mkdir ganglia_rpms

   b) Sincronizar archivos rpm del nodo maestro hacia el nodo de cálculo
      1) cd ganglia_rpms/
      2) rsync -av -P

root@master:/root/rpmbuild/RPMS/x86_64/*.rpm ./

1. Instalar paquetes rpm en nodo de cálculo

   a) Borrar archivo rpm de gmetad
      1) cd ganglia_rpms/
      2) rm -rf ganglia-gmetad*.rpm

   b) Instalar archivos rpm y sus dependencias
      1) yum install -y epel-release
      2) yum install -y libconfuse-devel
      3) yum install -y *.rpm

A.6. Configuración Ganglia (nodo maestro)

1. Editar el archivo /etc/ganglia/gmetad.conf

    a) Editar la línea:

        1) data_source “Nombre Cluster” 1 master.beowulf_uacm.mx

2. Editar el archivo /etc/ganglia/gmond.conf

    a) Modificar el archivo para las partes de la siguiente manera:

cluster{
name=“Nombre Cluster”
owner=“unespecified”
latlong=“unespecified”
url=“unespecified”
}
udp_send_channel{
# bind_hostname ...
# mcast_join ...
host = master.beowulf_uacm.mx
port = 8649
ttl = 1
}
(nota: comentar las variables mcast_join)
(nota: en caso de dudas revisar el video:)
udp_recv_channel {
# mcast_join = 239.2.11.71
port = 8649
# bind = 239.2.11.71
# retry_bind = true
# Size of the UDP buffer. If you are handling lots of metrics you really
# should bump it up to e.g. 10MB or even higher.
# buffer = 10485760
}
tcp_accept_channel {

port = 8649
# If you want to gzip XML output
# gzip_output = no
}
(guardar y salir)

1. Hacer:
   a) systemctl restart httpd
   b) systemctl enable httpd
   c) systemctl restart gmond
   d) systemctl enable gmond
   e) systemctl restart gmetad
   f) systemctl enable gmetad
2. ya puede revisar http://(IP del maestro)/ganglia/

A.7. Configuración Ganglia (nodo de cálculo)

1. En el nodo de cálculo copiar el archivo gmond.conf del nodo maestro
   a) scp root@master:/etc/ganglia/gmond.conf /etc/ganglia/
2. Habilitar puertos del firewall de centos 7
   a) firewall-cmd –permanent –zone=public –add-port=8649/udp
   b) firewall-cmd –permanent –zone=public –add-port=8649/tcp
   c) firewall-cmd –reload
3. Habilitar e iniciar el servicio gmond en el nodo de cálculo
   a) systemctl enable gmond
   b) systemctl start gmond

A.8. Configuración Ganglia Web (nodo maestro)

1. Navegar en la pagina http://ganglia.info y bajar el paquete rpm de ganglia-web version 3.1.1
2. Instalar paquete rpm (desde el directorio de descargas)
   a) yum install –y ganglia-web-3.1.1-1.noarch.rpm
3. Habilitar puertos del firewall de centos 7
   a) firewall-cmd –permanent –zone=public –add-service=http
   b) firewall-cmd –permanent –zone=public –add-port=8649/udp
   c) firewall-cmd –permanent –zone=public –add-port=8649/tcp
   d) firewall-cmd –permanent –zone=public –add-port=8651/tcp
   e) firewall-cmd –permanent –zone=public –add-port=8652/tcp
   f) firewall-cmd –reload
4. Iniciar servicios en el nodo maestro:
   a) systemctl restart httpd
   b) systemctl enable httpd
   c) systemctl restart gmond
   d) systemctl enable gmond
   e) systemctl restart gmetad
   f) systemctl enable gmetad
5. Navegar en la página web desde el nodo maestro http://master/ganglia y comprobar que los nodos aparecen gráficamente en la página web.

A.9. Instalación servidor NFS (nodo maestro)

1. Instalar las utilerías nfs
    a) yum install -y nfs-utils
2. Editar archivo /etc/idmapd.conf (descomentar):
    a) Domain = master
3. Instalar el system storage manager
    a) yum –y install system-storage-manager
    b) Listar particiones de discos
        1) ssm list
4. Habilitar servicios nfs-server y rpcbind
    a) systemctl start rpcbind nfs-server
    b) systemctl enable rpcbind nfs-server
5. Habilitar puertos firewall
    a) firewall-cmd –add-service=nfs –permanent
    b) firewall-cmd –reload
6. Verificar CONFIGURACIÓN nfs-server
    a) exportfs -a

NOTA: Realizar estos cambios desde una sesión ROOT del maestro.
NOTA: Revisar la pagina para habilitar los servicios debidos en todo el cluster: https://www.howtoforge.com/nfs-server-and-client-on-centos-7

1. lista las particiones del sistema
    a) df -h

2. Para desmontar y montar una unidad:

   a) umount /home/
   b) mount /home/

3. (man) makes an RPC call to an RPC server and reports what it finds.

   a) rpcinfo -p

4. (man): report information about volume groups

   a) vgs

5. (man): display attributes of a logical volume

   a) lvdisplay

6. (man): XFS filesystem incremental dump utility

   a) xfsdump
   b) yum info xfsdump

7. Crea una imagen del home si tienes usuarios activos

   a) xfsdump -l 0 -f /home.img /dev/cl/home
   b) ls /home.img
   c) ls -lh /home.img

8. Remueve el volumen lógico del home

   a) umount /home
   b) lvremove /dev/cl/home
   c) verifica con: vgs o df -h

9. Crea nuevos volúmenes lógicos

   a) lvcreate -L 100G -n home cl
   b) lvcreate -L 50G -n apps cl

   c) lvcreate -L 258.5G -n scratch cl
   d) NOTA: "cl" es un directorio dentro de "dev" verificar con "lvdisplay"
   e) Verifica la creación vgs

10. Crea el sistema de archivos para los volúmenes lógicos

   a) mkfs.xfs /dev/cl/home
   b) mkfs.xfs /dev/cl/apps
   c) mkfs.xfs /dev/cl/scratch

11. Restablecer el home

   a) mount /home
   b) ls -l /home
   c) xfsrestore -f /home.img /home
   d) ls -l /home

12. Editar el archivo (vim /etc/fstab) y agregar las líneas:

   a) /dev/mapper/cl-home /home xfs defaults 0 0
   b) /dev/mapper/cl-apps /share/apps xfs defaults 0 0
   c) /dev/mapper/cl-scratch /scratch xfs defaults 0 0

13. Crear los directorios sobre los cuales montar

   a) mkdir -p /share/apps
   b) mkdir /scratch
   c) mount /share/apps/
   d) mount /scratch/
   e) verificar los directorios con "df -h" o "vgs"

14. Crear o editar el archivo (vim /etc/exports) con las siguientes líneas

   a) /home 10.0.0.0/255.0.0.0(rw,async,no_root_squash)

   b) /share/apps 10.0.0.0/255.0.0.0(rw,async,no_root_squash)
   c) /scratch 10.0.0.0/255.0.0.0(rw,async,no_root_squash)

15. Activar e iniciar el demonio NFS

   a) systemctl start nfs
   b) systemctl status nfs

16. Ejecutar

   a) exportfs -a
   b) systemctl restart nfs
   c) systemctl status -l nfs
   d) exportfs
   e) env

17. Y revisar con "vgs" "df -h"

A.10. Instalación cliente NFS (nodos de cálculo)

1. Instalar utilerías NFS

   a) yum install -y nfs-utils

2. Editar el archivo (vim /etc/fstab) y agregar las líneas:

   a) master:/home /home nfs rsize=8192, wsize=8192, timeo=14, intr 0 0
   b) master:/scratch /scratch nfs rsize=8192, wsize=8192, timeo=14, intr 0 0
   c) master:/share/apps /share/apps nfs rsize=8192, wsize=8192, timeo=14, intr 0 0
   d) NOTA: Comentar o eliminar el antiguo home

3. Iniciar los servicios

    a) systemctl start nfs
    b) systemctl enable nfs
    c) systemctl status nfs

4. Existen los directorios ???

    a) find / -name home
    b) find / -name scratch
    c) find / -name share

5. Sí sí, desmontar carpetas

    a) umount /home
    b) umount /scratch
    c) umount /share/apps
    d) NOTA: Eliminar el home:
        1) lvremove /dev/mapper/cl-home

6. Crear directorios si no existen

    a) mkdir /home
    b) mkdir /scratch
    c) mkdir -p /share/apps

7. Montar los directorios

    a) mount master:/home /home
    b) mount master:/scratch /scratch
    c) mount master:/share/apps /share/apps

8. Verificar se hayan realizado los cambios

    a) df -h
    b) vgs

c) lsblk -f

9. NOTA: En este punto el orden en que se apagan/encienden los nodos es importante:

   a) Aparga:

      1) primero nodos, ultimo maestro

   b) Encender:

      1) *a′* primero maestro, luego los nodos (de lo contrario marcara errores en el inicio del SO)

A.11. Instalación TORQUE (nodo maestro)

1. Revisar tutorial

   a) www.youtube.com/watch?v=SCHoEjson_A&t=308s

2. Habilitar puertos firewall

   a) firewall-cmd –permanent –zone=public –add-port=15001/tcp

   b) firewall-cmd –permanent –zone=public –add-port=15002/tcp

   c) firewall-cmd –permanent –zone=public –add-port=15003/tcp

   d) firewall-cmd –reload

   e) firewall-cmd –list-all

3. Instalar dependencias

   a) yum install -y libtool openssl-devel libxml2-devel boost-devel gcc gcc-c++

4. Navegar a la pagina http://www.adaptivecomputing.com

   a) En el menu support -> download center -> open source products
   b) Bajar el archivo torque-x.x.x.tar.gz

5. Descomprimir archivo torque-x.x.x.tar.gz

   a) tar -zxvf torque-x.x.x.tar.gz

6. Configurar y compilar

   a) cd torque-6.1.0/
   b) ./configure
   c) make
   d) make install

7. copiar los siguientes servicios al sistema

   a) cp contrib/init.d/pbs_server /etc/init.d/pbs_server
   b) cp contrib/init.d/pbs_sched /etc/init.d/pbs_sched
   c) cp contrib/init.d/trqauthd /etc/init.d/trqauthd

8. habilitar e iniciar los servicios

   a) chkconfig pbs_server on
   b) chkconfig pbs_sched on
   c) chkconfig trqauthd on
   d) service trqauthd start

9. Habilitar librerias en el sistema

   a) echo “master.beowulf_uacm.mx”> /var/spool/torque/server_name
   b) echo “usr/local/lib”>/etc/ld.so.conf.d/torque.conf
   c) ldconfig
   d) ./torque.setup root
      1) (yes)

10. Agregar los nodos
    a) vim /var/spool/torque/server_priv/nodes
        1) peon221.beowulf_uacm.mx np=2 peon1
        2) peon222.beowulf_uacm.mx np=2 peon2
        3) ...
            a′ NOTA: np es el número de cpus del nodo y la última parte es una propiedad que describa al nodo
11. iniciar los servicios
    a) service pbs_server start
    b) service pbs_sched start
12. finalmente construimos los paquetes para los nodos
    a) make packages
13. ***(Actualizar)
14. Editar archivo torque.setup
    a) En la parte final, # create default queue, cambiar "batch" por "nombre cluster"
        1) qmgr -c 'create queue beowulf_uacm'
        2) qmgr -c 'set queue beowulf_uacm queue_type = execution'
        3) qmgr -c 'set queue beowulf_uacm started = true'
        4) qmgr -c 'set queue beowulf_uacm enabled = true'
        5) qmgr -c 'set queue beowulf_uacm resources_default.walltime = 720:00:00'
        6) qmgr -c 'set queue beowulf_uacm resources_default.nodes = 4'
           (NOTA: nodos disponibles)
15. Ejecutar script de inicio de torque pbs_server
    a) ./torque.setup root
        1) (yes)

A.12. Instalación TORQUE (nodos de cálculo)

1. Habilitar reglas de firewall
   a) firewall-cmd –permanent –zone=public –add-port=15001/tcp
   b) firewall-cmd –permanent –zone=public –add-port=15002/tcp
   c) firewall-cmd –permanent –zone=public –add-port=15003/tcp
   d) firewall-cmd –reload
   e) firewall-cmd –list-all
2. Instalar dependencias:
   a) yum install -y libtool openssl-devel libxml2-devel boost-devel gcc gcc-c++
3. copiar el directorio torque del maestro a los nodos
   a) rsync -av -P root@master:torque-6.1.1.1 ./
4. entrar y hacer
   a) cd torque-6.1.1.1/
   b) ./torque-package-mom-linux-x86_64.sh –install
   c) ./torque-package-clients-linux-x86_64.sh –install
5. verificar que existe la ruta “/usr/local/lib” en el siguiente archivo:
   a) cat /etc/ld.so.conf.d/torque.conf
      1) sí no está entonces agregarla:
         a′ echo ’/usr/local/lib’>/etc/ld.so.conf.d/torque.conf
   b) luego ejecutar:

1) ldconfig

6. Luego copiar los demonios a los directorios:
   a) cp contrib/init.d/pbs_mom /etc/init.d/
   b) cp contrib/init.d/trqauthd /etc/init.d/
   c) ls /etc/init.d/
   d) Habilitarlos:
      1) systemctl enable trqauthd.service
      2) systemctl enable pbs_mom.service
7. Iniciar el servicio
   a) systemctl start trqauthd.service
      1) SI NO SE INICIA, validar hostname y agregar al script, usar:
      2) hostname
      3) Y agregar a: vim /etc/sysconfig/network
         a′ "HOSTNAME=peon221.beowulf_uacm.mx"
8. Agregar, si no existen, las siguientes variables al archivo (cat /var/spool/torque/mom_priv/config):
   a) echo '$pbsserver master'> /var/spool/torque/mom_priv/config
   b) echo '$logevent 225'> > /var/spool/torque/mom_priv/config
9. iniciar el servicio
   a) systemctl start pbs_mom.service
10. verificar el estado de los servicios
    a) systemctl status trqauthd.service
    b) systemctl status pbs_mom.service
11. Verificar la activación en el nodo maestro:
    a) Pbsnodes

A.13. Instalación PDSH (nodo maestro)

1. PDSH sirve para ejecutar instrucciones en todos los nodos al mismo tiempo
2. Navegar a la página:
   - *a*) https://code.google.com/archive/p/pdsh/downloads
   - *b*) Bajar la ultima versión pdsh-2.29.tar.bz2
3. Instalar pdsh
   - *a*) tar –jxvf pdsh-2.29.tar.bz2
   - *b*) cd pdsh-2.29
   - *c*) ./configure –with-ssh –without-rsh
   - *d*) make
   - *e*) make install
   - *f*) pdsh -V
4. Crear llave ssh para usuario root
   - *a*) ssh-keygen
     1) Click Enter a los siguientes tres que aparecen
        - *a′*) (/root/.ssh/id_rsa) (enter)
        - *b′*) passphrase: (enter)
        - *c′*) passphrase again: (enter)
5. Verificar creación de llaves
   - *a*) ls -l $HOME/.ssh/id*
6. Copiar llave ssh a nodos de cálculo
   - *a*) ssh-copy-id peonXYZ
   - *b*) ssh peonXYZ
   - *c*) exit

d) NOTA: (repetir tantos como nodos activos halla)

7. Probar comando pdsh

   a) pdsh -w peonXY[1-5] “hostname”

8. Crear directorio PDSH

   a) mkdir /etc/pdsh

   b) Crear archivo de lista de nodos (vim /etc/pdsh/hosts) y agregar:

      1) peonXYZ
      2) NOTA: (Agregar tantos como nodos activos halla)

9. Instalar paquete openssh-askpass en todos los nodos

   a) Nodo Maestro

      1) yum install -y openssh-askpass

   b) Nodos Esclavo

      1) pdsh -w peonXY[1-5] “yum install -y openssh-askpass”

10. Habilitar lista de nodos

    a) export WCOLL=/etc/pdsh/hosts

11. Probar pdsh con lista de nodos

    a) pdsh “hostname”

A.14. Configuración de usuarios (nodo maestro)

1. Creamos un usuario:

   a) useradd -m nombre_usuario

2. Verificamos el ID de usuario

   a) cat /etc/passwd | grep nombre_usuario

3. Agregamos al usuario al grupo en todos los nodos

   a) pdsh "groupadd -g 1001 nombre_usuario"

   b) pdsh "useradd -u 1001 -g 1001 -m nombre_usuario"

4. Creamos un carpeta para almacenar los passwords de los usuarios

   a) vi /home/usuarios/password.nombre_usuario

   b) creamos un archivo por cada usuario y almacenamos en él dos veces su password

   c) NOTA: solo el administrador puede saber los passwords

5. Asignamos el archivo como password del usuario

   a) Nodo maestro:

      1) passwd nombre_usuario < /home/usuarios/password.nombre_usuario

   b) Nodos esclavo:

      1) pdsh "passwd nombre_usuario < /home/usuarios/password.nombre_usuario"

6. Iniciamos sesión como root del nuevo usuario

   a) su - nombre_usuario

7. Generamos su llave

   a) ssh-keygen -t rsa (teclear enter en todas las opciones)

8. Copiamos la llave a los nodos

   a) ssh-copy-id -i .ssh/id_rsa.pub peon221

   b) ssh peonXYZ1

   c) ssh peonXYZ2

d) etc...

9. Comprobamos su acceso a los nodos con un pdsh

   a) pdsh “hostname”

   b) exit

(FIN)

Apéndice B

Códigos Arduino

En el presente apéndice se muestra el código que se implementó para cada una de las estaciones meteorológicas en Arduino. Dicho código fue una implementación de varias fuentes por cada uno de los dispositivos conectados a ellas, sensor de humedad [29], presión barométrica [30], densidad de polvo [31], velocidad de viento [32] y tarjeta micro SD [33].

B.1. Código Arduino de la Estación Meteorológica

```
//Bibliotecas
#include <dht.h> // humedad y temperatura C
#include <Adafruit_BMP085.h> // presion y temperatura
#include <Wire.h> // complemento BMP085
#include <SPI.h>
#include <SD.h>
#include <DS1302.h> // Real Time Clock (RTC)

#define DHT11_PIN 6
#define HALL_PIN 2
#define LED_ERROR_PIN 10

// Instancias de Objetos
dht DHT;
```

```
Adafruit_BMP085 bmp;

// Config Sensor Polvo GP2Y10
int measurePin = 0; //sensor de polvo al pin A0
int ledPower = 7;   //sensor de polvo al pin B7
int samplingTime = 280;
int deltaTime = 40;
int sleepTime = 9680;
float voMeasured = 0;
float calcVoltage = 0;
float dustDensity = 0;

// Config Sensor Anemometro
volatile unsigned long rev;
volatile unsigned long revinicial;
volatile unsigned long revfinal;
volatile unsigned long tinicial;
volatile unsigned long tfinal;
volatile unsigned long freq;
volatile unsigned long rpm;
volatile byte ppin;

//SD
const int chipSelect = 4;

// --------------------------- TIME (RTC)
namespace {
  // Set the appropriate digital I/O pin connections.
  // These are the pin assignments for the Arduino
  // as well for as the DS1302 chip.
  // See the DS1302 datasheet:
// http://datasheets.maximintegrated.com/en/ds/DS1302.pdf
  const int kCePin   = 3;  // Chip Enable
  const int kIoPin   = 5;  // Input/Output
  const int kSclkPin = 9;  // Serial Clock

  // Create a DS1302 object.
  DS1302 rtc(kCePin, kIoPin, kSclkPin);
```

```
String dayAsString(const Time::Day day) {
  switch (day) {
    case Time::kMonday: return "Lunes";
    case Time::kTuesday: return "Martes";
    case Time::kWednesday: return "Miercoles";
    case Time::kThursday: return "Jueves";
    case Time::kFriday: return "Viernes";
    case Time::kSaturday: return "Savado";
    case Time::kSunday: return "Domingo";
  }
  return "(unknown day)";
}

void printTime() {
  // Get the current time and date from the chip.
  Time t = rtc.time();

  // Name the day of the week.
  const String day = dayAsString(t.day);

  // Format the time and date and insert into the
  // temporary buffer.
  char buf[50];
  snprintf(buf, sizeof(buf),
  "%s %04d-%02d-%02d %02d:%02d:%02d",
           day.c_str(),
           t.yr, t.mon, t.date,
           t.hr, t.min, t.sec);

  // Print the formatted string to serial so we
  // can see the time.
  Serial.println(buf);
}

String getTime() {
  Time t = rtc.time();
  const String day = dayAsString(t.day);

  // Format the time and date and insert into the
```

```
    // temporary buffer.
    char buf[50];
    snprintf(buf, sizeof(buf),
    "%s %04d-%02d-%02d %02d:%02d:%02d",
             day.c_str(),
             t.yr, t.mon, t.date,
             t.hr, t.min, t.sec);

    return(buf);
  }

  String getFecha() {
    Time t = rtc.time();
    const String day = dayAsString(t.day);
    char buf[15];
    snprintf(buf, sizeof(buf), "%04d-%02d-%02d",
     t.yr, t.mon, t.date);
    return(buf);
  }

  String getHora() {
    Time t = rtc.time();
    const String day = dayAsString(t.day);
    char buf[10];
    snprintf(buf, sizeof(buf), "%02d:%02d:%02d",
     t.hr, t.min, t.sec);
    return(buf);
  }
}  // namespace
// -------------------------------- TIME END (RTC)

void errorSD(int k, int ms){
  int i;
  for( i=0 ; i<=k ; i++){
    digitalWrite(LED_ERROR_PIN, HIGH);
    delay(ms);
    digitalWrite(LED_ERROR_PIN, LOW);
    delay(ms);
  }
```

```
  Serial.println("Error en SD");
}

void setup(){// ------------------------------ SETUP
  attachInterrupt(0, funrev, RISING);
  //registra interrupciones desde el hall
  pinMode(HALL_PIN,INPUT);
  // lectura de hall por el digital 2

  Serial.begin(9600);

  if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP180 sensor,
    check wiring!");
    while (1) {}
  } //bmp180

  pinMode(ledPower,OUTPUT); // polvo GP2Y10

  // ------ Establece Coneccion con la SD -----
  pinMode(LED_ERROR_PIN, OUTPUT);
  if (!SD.begin(chipSelect)){ // si error, entonces:
    errorSD(30, 1000);
    return;
  }

  // -------------- time
  // Se inicializa el contador de Fecha y de Hora.
  // Se monta/compila una vez en la placa y luego se
  // "comenta" y se vuelve a montar/compilar
  // NOTA: Desconectar el cable del pin VCC
/*
  // Initialize a new chip by turning off write
  // protection and clearing the clock halt flag.
  // These methods needn't always be called.
  // See the DS1302 datasheet for details.
  rtc.writeProtect(false);
  rtc.halt(false);
```

```
  // Make a new time object to set the date and time.
  // Formato Time t( aaaa-mm-dd , hh:mm:ss , dia )
  Time t(2017, 7, 23, 3, 57, 00, Time::kSaturday);

  // Set the time and date on the chip.
  rtc.time(t);
  rtc.writeProtect(true);
*/
  // -------------- time end

  rev = 0; //resetea las rev a cero
  tinicial = 0; //contador de tiempo a cero

}

void loop(){ // ------------------------------ LOOP
  // Lectura de Datos
  int chk = DHT.read11(DHT11_PIN); // se inicializa
  /* ALERTA: no se usa la variable chk */

  /* chequeo del dth11
  Serial.print("DHT11: ");
  switch (chk) {
    case DHTLIB_OK:
     Serial.print("OK,\t"); break;
    case DHTLIB_ERROR_CHECKSUM:
     Serial.print("Checksum error,\t"); break;
    case DHTLIB_ERROR_TIMEOUT:
     Serial.print("Time out error,\t"); break;
    case DHTLIB_ERROR_CONNECT:
     Serial.print("Connect error,\t"); break;
    case DHTLIB_ERROR_ACK_L:
     Serial.print("Ack Low error,\t"); break;
    case DHTLIB_ERROR_ACK_H:
     Serial.print("Ack High error,\t"); break;
    default: Serial.print("Unknown error,\t"); break;
  }
  */
```

```
// Lectura de variables
float hum1 = DHT.humidity;
float tmp1 = DHT.temperature;
float tmp2 = bmp.readTemperature();
float ps = bmp.readPressure();
float pps = 1./101325.*ps;
/* ALERTA: esta variable no se usa */

// Sensor Polvo
digitalWrite(ledPower,LOW);
// apaga el pin B7 (power on the LED)
delayMicroseconds(samplingTime);
voMeasured = analogRead(measurePin);
// lectura desde pin A0 (read the dust value)
delayMicroseconds(deltaTime);
digitalWrite(ledPower,HIGH);
// enciende el led en B7 (turn the LED off)
delayMicroseconds(sleepTime);

// 0V - 5V mapped to 0 - 1024 integer values
// recover voltage
calcVoltage = voMeasured * (5.0 / 1024.0);
/* ALERTA: Verificar la corriente en la que esta
* conectado el sensor
*/

// linear eqaution taken from:
// http://www.howmuchsnow.com/arduino/airquality/
// Chris Nafis (c) 2012
dustDensity = 0.17 * calcVoltage - 0.1;
//calibración de la densidad
float dens = 1000.*sqrt(dustDensity*dustDensity);
// rectificacion de parametro

// Config Sensor Anemometro
revinicial = rev; //congela rev
tinicial = millis();
ppin = digitalRead(HALL_PIN);
delay(1000);
```

```
tfinal = millis();
revfinal = rev;
freq = 1000.*(revfinal-revinicial)/(tfinal-tinicial)/2.;
/* ALERTA: revisar la ecuacion */

// rpm = freq*60.; //rev por minuto
float frq = (float)freq;

// Desplegar al serial (pantalla)
Serial.print("[ ");
Serial.print(getTime()); Serial.print(" ] ");
Serial.print(hum1, 1);  Serial.print(" "); //  [%]
Serial.print(tmp1, 1);  Serial.print(" "); //  [C]
Serial.print(tmp2, 2);  Serial.print(" "); //  [C]
Serial.print(ps,2);     Serial.print(" "); //  [Pa]
Serial.print(dens,4);   Serial.print(" "); //  [mg/m3]
Serial.print(frq, 1);
Serial.println(" "); //  [rev/minuto]

// Escribir en memoria SD
/**
 * Formato para las columnas dentro del archivo
 * ARCHIVO.txt={fecha, hora, estacionNo, edificio,
 * posision, altura, hum1, tmp1, tmp2, ps, dens, frq}
 * ARCHIVO.txt={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
 */
float arr[6]={hum1,tmp1,tmp2,ps,dens,frq};
//arreglo con las variables
File dataFile = SD.open("EM-07.txt", FILE_WRITE);
// crea y escribe sobre el archivo

// Si el fichero es correcto escribimos en el.
if (dataFile) {
  dataFile.print(getFecha());
  dataFile.print(" ");
  dataFile.print(getHora());
  dataFile.print(" ");
  dataFile.print("e7");
  // Estacion No. = e1, e2, e3, e4, e5, e6, e7, e8, e9
```

```
    dataFile.print(" ");
    dataFile.print("C");
    // Coordenada X = Edificio = A, B, C
    dataFile.print(" ");
    dataFile.print("Y1");
    // Coordenada Y = Longitud = Y1, Y2, Y3
    dataFile.print(" ");
    dataFile.print("P0");
    // Coordenada Z = Altura = P0, P1, P2
    dataFile.print(" ");

    // Escribe Todos los Datos, en una linea:
    int k;
    for ( k=0 ; k<6 ; k=k+1 ) {
      dataFile.print(arr[k]);
      dataFile.print(" ");
    }
    dataFile.println(" ");
    dataFile.close();
  }else{
    // mostrar ERROR de SD por led arduino
    errorSD(5,100);
  }

  rev=0;  // reset a cero el rev
  delay(2000);
}

//funciones
void funrev(){
  rev++; //cada rev se aplica dos veces
}
```

FIN

Apéndice C

Base de Datos en PostgreSQL

C.1. Creación de Base de Datos

– 1) Crear tabla que contendrá todos los registros de todas las estaciones
– NOTA: Previamente se seleccionaron los registros de las 00:00:00 hasta las 23:59:59,
– equivalente a 24 hrs y se eliminaron columnas innecesarias.

```
DROP TABLE IF EXISTS datos24h;
CREATE TABLE datos24h (
humedad NUMERIC,
tmp1 NUMERIC,
tmp2 NUMERIC,
presion NUMERIC,
polvo NUMERIC,
viento NUMERIC,
estacion VARCHAR,
hora TIME, – permite el formato 'HH:MI:SS'
id NUMERIC
);
```

– 1.1) Importar Registros (csv) a la TABLA

```
TRUNCATE TABLE datos24h; – limpiamos la tabla
COPY datos24h
FROM 'C:\ALMACEN_DATOS\est1.csv'
```

```
DELIMITER ',' CSV HEADER;
COPY datos24h
FROM 'C:\ALMACEN_DATOS\est2.csv'
DELIMITER ',' CSV HEADER;
– hasta la nueve
COPY datos24h
FROM 'C:\ALMACEN_DATOS\est9.csv'
DELIMITER ',' CSV HEADER;

SELECT * FROM datos24h; – verificamos la tabla

– 1.2) crear respaldo de la tabla
COPY datos24h
TO 'C:\ALMACEN_DATOS\resp_tab_datos24h.csv'
DELIMITER ',' CSV HEADER;
```

C.2. Promedios por minuto

```
– 2) Crear una nueva tabla por cada estación para importar los
promedios por minuto
– 2.1) Crea las nuevas tablas e1, e2, ... , e9
– Nota: tem1 = Abreviación de Tabla Estación Meteorológica 1
DROP TABLE IF EXISTS tem1;
CREATE TABLE tem1 (
hora TEXT,
humedad NUMERIC,
tmp1 NUMERIC,
tmp2 NUMERIC,
presion NUMERIC,
polvo NUMERIC,
viento NUMERIC
);

– 2.2) Ingresa los registros promediados a las nuevas tablas por
estación
INSERT INTO tem1
SELECT
taux.hh ||':'|| taux.mm AS hora,
```

```
round(avg(taux.humedad),2) AS humedad,
round(avg(taux.tmp1),2) AS tmp1,
round(avg(taux.tmp2),2) AS tmp2,
round(avg(taux.presion),2) AS presion,
round(avg(taux.polvo),2) AS polvo,
round(avg(taux.viento),2) AS viento
FROM (
SELECT
EXTRACT(HOUR FROM hora) AS hh,
EXTRACT(MINUTE FROM hora) AS mm,
EXTRACT(SECOND FROM hora) AS ss,
humedad, tmp1, tmp2, presion, polvo, viento
FROM datos24h
WHERE estacion='B'
) AS taux
GROUP BY taux.hh, taux.mm
ORDER BY taux.hh, taux.mm;

– Verificamos los registros para cada estación
SELECT * FROM tem1;

– 2.3) Exportar las tablas a archivos CSV (como respaldo de la DB)
COPY tem1 TO 'C:\ALMACEN_DATOS\Ttem100m.csv'
DELIMITER ',' CSV HEADER;
```

C.3. Tablas por variable meteorológica

```
– Crear tablas por tipo de variable e importar datos
– Nota: Como ejemplo se muestra el procedimiento para una de las
variables, Temperatura.
– este procedimiento se repite para cada una de nuestras variables.

– 1) Creamos las nuevas tablas por tipo de variable
DROP TABLE IF EXISTS temperatura;
CREATE TABLE temperatura (
hora TEXT,
ce1 NUMERIC, – ceX, abreviacion de Columna Estacion X
ce2 NUMERIC,
ce3 NUMERIC,
```

```
ce4 NUMERIC,
ce5 NUMERIC,
ce6 NUMERIC,
ce7 NUMERIC,
ce8 NUMERIC,
ce9 NUMERIC
);

– 2) Importamos las variables de temperatura a la nueva tabla
TRUNCATE temperatura;
INSERT INTO temperatura (hora, ce1) SELECT tem1.hora,
tem1.tmp2 FROM te1;
UPDATE temperatura SET ce2 = tem2.tmp2 FROM ec WHERE
temperatura.hora=tem2.hora;
UPDATE temperatura SET ce3 = tem3.tmp2 FROM ed WHERE
temperatura.hora=tem3.hora;
– etc... continuar así hasta la ultima estacion meteorologica tem9
– Observación: se agrega la columna hora de cualquier tabla, ya que
es la misma para todas.

– Nota: en caso de que hayan fallado una estacion, ésta se calcula con
el valor de sus vecinas.
– Ejemplo, calcular los registros de las estacion 1:

/*
La configuración espacial y tablas faltantes*
ED-A: e7 e8 e9
ED-B: e4 e5 e6
ED-C: e1* e2 e3
e1 = (e2+e4)/2
*/

UPDATE temperatura
SET ce1=ROUND((ce2+ce4)/2,2),
WHERE hora!=";

SELECT * FROM temperatura; – verificamos la actualizacion

– 3) Generamos respaldo de la tabla
COPY temperatura
```

```
TO 'C:\ALMACEN_DATOS\respaldo_temperatura.csv'
DELIMITER ',' CSV HEADER;
```

Apéndice D

Implementación de Minería de Datos en Lenguaje R

Para los ejemplos a continuación sobre la implementación de las funciones en R para el análisis de datos, se tomará como base la variable ambiental temperatura, dejando al lector implementar las demás variables dado que las funciones que se emplean serán las mismas para cada una.

Para el primer ejemplo con el uso de la librería "gcc", se verá como importar datos desde un archivo CSV. Y para el segundo ejemplo con la librería "geoR", se verá como importar registros desde la Base de Datos en PostgreSQL.

D.1. Gráficos estadísticos y uso de librería gcc

```
# importar las librerías
library(qcc) # métodos estadísticos
library(lubridate) # formato de horas

# 1) importamos la tabla
misDatos <- read.csv(
file='C:/ALMACEN_DATOS/respaldo_temperatura.csv',
header=TRUE, sep=',')
```

```
# renombramos las columnas
colnames(misDatos) <- c('hora','e1','e2','e3','e4',
'e5','e6','e7','e8','e9')

# 2) Sacamos las estadísticas básicas de todas las estaciones
summary(misDatos[,2:10])

# 3) convertir el formato de tiempo
aux_h <- hm(misDatos$hora) # convertimos el formato
misDatos$hora <- aux_h # reasignamos la columna

# 4) Plot grafico media y desv estándar
medias<-qcc(data = misDatos, type = 'xbar', title = 'Temperatura
VS Tiempo',
Xlab='[ t=24h 1̃440min]', ylab='[ °C ]', nsigmas = 4)

# mostramos el contenido de la variable medias
medias

# 5) Crear grafico para las 9 estaciones
ngraficas<-ncol(misDatos)

# Obtenemos el rango de X e Y
xrange <- range(misDatos)
yrange <- range(misDatos$cea)

# Generamos la ventana para los gráficos
plot(misDatos$hora, misDatos[,2],
main='Temperatura VS Tiempo', xlab='[ t=86400s (24hrs) ]',
ylab='[ °C ]', col='red', type = 'l', ylim = c(8, 26))
colors <- rainbow(ngraficas)
linetype <- c(1:ngraficas)
plotchar <- seq(15,15+ngraficas,1) # max25

# Agregamos los demás gráficos en la ventana
for (i in 3:ngraficas) {
lines(misDatos$hora, misDatos[,i], type='l', lwd=1.5,
lty=linetype[i], col=colors[i], pch=plotchar[i])
}

# Agregamos la etiqueta con información de las líneas
```

```
legend(xrange[1], yrange[2], 2:ngraficas-1, cex=0.8,
col=colors, pch=plotchar, lty=linetype, title='Estación')
```

D.2. Método de interpolación Kriging y uso de librería geoR

```
# importar las librerías
library(geoR) #install.packages('geoR')
library(DBI) #install.packages('DBI')
library(RPostgreSQL) #install.packages('RPostgreSQL')

# 1) Realizamos la conexión a la base de datos
conexionBD = dbConnect(PostgreSQL(), user='postgres',
password='manager', dbname='postgres')

# 2) Consultamos e importamos los datos de temperatura
datosSQL = dbGetQuery (conexionBD, 'SELECT * FROM temperatura WHERE hora="14:30";')

# podemos ver la tabla con
View(datosSQL)

# 3) crear el objeto GeoData
# 3.1) Crear una matriz con la información de la configuración espacial y los datos de las estaciones
# Configuración espacial de las estaciones:
# ED-A: e7(0,2), e8(1,2), e9(2,2)
# ED-B: e4(0,1), e5(1,1), e6(2,1)
# ED-C: e1(0,0), e2(1,0), e3(2,0)

m<- matrix(1:27,nrow=9,ncol=3)
x<-c(0, 1, 2, 0, 1, 2, 0, 1, 2)
y<-c(0, 0, 0, 1, 1, 1, 2, 2, 2)
m[,1]<-x
m[,2]<-y
for (i in 1:9) {
m[i,3]<-datosSQL[,i+1]
}

# verificamos la matriz m

# 3.2) Convertir la Matriz a objeto GEODATA
geodatos <- as.geodata(m, coords.col = 1:2, data.col = 3)
```

```
# 4) Proceso de KRIGING:
# 4.1) Creamos la Rejilla regular de 200x200 para el método de interpolación
xx <- seq(0, 2, l = 200) ## sec( min-X , max-X , long=N )
yy <- seq(0, 2, l = 200) ## sec( min-Y , max-Y , long=N )
sub.malla <- expand.grid(x = xx, y = yy)

# 4.2) aplicamos la función de Kriging
aux_krg <- krige.conv (geodatos, loc=sub.malla,
krige=krige.control(cov.pars=c(2, .5)))
#names(aux_krg)

# creamos una ventana dividida para las graficas
layout(matrix(1:2,1,2)) # horizontal

# 5) Graficamos Isosuperficie 2D
image(aux_krg, col=ramp.col(
c('blue3','blue','cyan','green','yellow','orange','red','red4')))
title('Temperaturas °C')
points(geodatos$coords, pch=17) # añadir posiciones de estaciones
contour(aux_krg,add=T) # añadir isolineas

# 6) Graficamos Isosuperficie 3D
library(plot3D) #install.packages('plot3D')
if(!require(plot3D))
stop('Required pakage 'plot3D' not installed.')

persp3D(xx, yy, matrix(aux_krg$predict, nrow = length(xx)),
theta=-60, phi=30)
title('Temperaturas °C')
# Nota: theta y phi son los grados de rotación en la gráfica
```

Bibliografía

[1] Cambridge University Press. Significado de cluster, Última revisión diciembre de 2018. https://dictionary.cambridge.org/es/diccionario/ingles/cluster.

[2] Oxford University Press. Significado de cluster, Última revisión diciembre de 2018. https://en.oxforddictionaries.com/definition/cluster.

[3] Revista Digital Universitaria. ¿qué es un cluster?, Última revisión diciembre de 2018. http://www.revista.unam.mx/vol.4/num2/art3/cluster.htm.

[4] Computer History Museum. Internet history 1962 to 1992, Última revisión diciembre de 2018. http://www.computerhistory.org/internethistory.

[5] Carlos Enrique Rodas G. Tutorial para realizar un cluster, Última revisión diciembre de 2018. https://carlos8rg.wordpress.com/2008/08/04/tutorial-para-realizar-un-cluster/.

[6] Andrew S Tanenbaum. *Structured computer organization*. Pearson Education India, 2016.

[7] Ganglia Community Developers. What is ganglia?, Última revisión diciembre de 2018. http://ganglia.info/.

[8] GitHub community. Ganglia development team, Última revisión diciembre de 2018. https://github.com/ganglia.

[9] Red Hat. Chapter 6. configuring an nfs cluster service, Última revisión diciembre de 2018.

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/configuration_example_-_nfs_over_gfs/nfs_gfs_service_configuration.

[10] Adaptive Computing. Torque resource manager, Última revisión diciembre de 2018. http://www.adaptivecomputing.com/products/torque/.

[11] Adaptive Computing. Torque administrator guide overview, Última revisión diciembre de 2018. http://docs.adaptivecomputing.com/torque/4-2-7/Content/topics/0-intro/guideOverview.htm.

[12] MundoAVR. Arduino el documental, Última revisión diciembre de 2018. https://www.youtube.com/watch?v=mltWc9_C9gs.

[13] Arduino. What is arduino?, Última revisión diciembre de 2018. https://www.arduino.cc/en/Guide/Introduction.

[14] Luis Llamas. Medir temperatura y humedad con arduino y sensor dht11-dht22, Última revisión diciembre de 2018. https://www.luisllamas.es/arduino-dht11-dht22/.

[15] Carlos Escobar. Bmp180, sensor barométrico con arduino, Última revisión diciembre de 2018. https://hetpro-store.com/TUTORIALES/bmp180-sensor-barometrico/.

[16] SHARP. Gp2y1010au0f, Última revisión diciembre de 2018. https://www.sparkfun.com/datasheets/Sensors/gp2y1010au_e.pdf.

[17] Arduino Modules. Ky-003 hall magnetic sensor module, Última revisión diciembre de 2018. https://arduinomodules.info/ky-003-hall-magnetic-sensor-module/.

[18] Google. Google maps,
Última revisión diciembre de 2018.
https://www.google.com.mx/maps.

[19] Oscar Nigro, Daniel Xodo, Gabriel Corti, and Damián Terren. Kdd (knowledge discovery in databases): Un proceso centrado en el usuario. In *VI Workshop de Investigadores en Ciencias de la Computación*, 2004.

[20] Maximiliano Silva. Minería de datos y descubrimiento del conocimiento,
Última revisión diciembre de 2018.
http://exa.unne.edu.ar/informatica/SO/
Mineria_de_Datos_y_KDD.pdf.

[21] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996.

[22] The R Foundation. What is r?,
Última revisión diciembre de 2018.
https://www.r-project.org/about.html.

[23] Sergio Santana Sepúlveda, Efraín Mateos Farfán, et al. El arte de programar en r: un lenguaje para la estadística, 2014.

[24] RStudio. Why rstudio?,
Última revisión diciembre de 2018.
https://www.rstudio.com/about/.

[25] Richard Webster and Margaret A Oliver. *Geostatistics for environmental scientists*. John Wiley & Sons, 2007.

[26] Rubén Fernández Casal. Introducción a la geoestadística con 'geor', Última revisión diciembre de 2018.
https://rubenfcasal.github.io/post/introducci

[27] Paulo J. Ribeiro Jr and Peter J. Diggle. Package 'geor',
Última revisión diciembre de 2018.
https://cran.r-project.org/web/packages/geoR/geoR.pdf.

[28] Roberto Carro and Daniel A González Gómez. Control estadístico de procesos. 2012.

[29] Antony García González. Dht11: Sensor de humedad/temperatura para arduino, Última revisión diciembre de 2018. http://panamahitek.com/dht11-sensor-de-humedadtemperatura-para-arduino/.

[30] Tony DiCola. Adafruit-bmp085-library, Última revisión diciembre de 2018. https://github.com/adafruit/Adafruit-BMP085-Library/blob/master/examples/BMP085test/BMP085test.ino.

[31] Arduino Tutorials. Standalone: Sharp dust sensor, Última revisión diciembre de 2018. http://arduinodev.woofex.net/2012/12/01/standalone-sharp-dust-sensor/.

[32] GitHub community. Ky 003 hall sensor, Última revisión diciembre de 2018. https://github.com/R2D2-2017/R2D2-2017/wiki/KY-003-Hall-sensor.

[33] Enrique. Cómo leer y escribir datos en la tarjeta sd de arduino, Última revisión diciembre de 2018. http://www.educachip.com/como-leer-y-escribir-datos-en-la-tarjeta-sd-de-arduino/.